

PRACA INŻYNIERSKA DYPLOMOWA

REGULATOR OBROTÓW SILNIKA BLDC

Maksymilian Piechocki

Promotor: dr inż. Stanisław Gardecki

Wydział Automatyki, Robotyki i Elektrotechniki
Instytut Automatyki, Robotyki i Inżynierii Informatycznej
Politechniki Poznańskiej

Poznań 2020

Streszczenie

Praca dotyczy procesu projektowania, wykonania, a następnie oprogramowania i testów elektronicznego regulatora obrotów silnika BLDC, używanego w robotach latających. Zawiera wiedzę teoretyczną i praktyczną, dotyczącą sterowania silnikiem BLDC, projektowania układów energoelektronicznych z transtorami MOSFET, oraz oprogramowania mikrokontrolera STM32F4 w języku C.

Abstract

The following thesis deals with the process of designing, manufacturing and then programming and testing of electronic speed controller of a flying robot BLDC motor. It contains knowledge, both theoretical and practical, concerning BLDC motor controll, design of power electronics with MOSFET transistors and programming of STM32F4 microcontroller in C programming language.

Spis treści

1	Wstęp	5
1.1	Cel i zakres pracy	5
1.2	Założenia projektowe	5
1.3	Struktura pracy	6
2	Teoria dotycząca sterowania silnikiem BLDC	7
2.1	Budowa bezszczotkowego silnika prądu stałego	7
2.2	Schemat zasilania faz	8
2.3	Ograniczenie natężenia przepływającego prądu	9
2.4	Sposób wykrycia przejścia przez zero	10
3	Główny układ energoelektroniczny	13
3.1	Zasilanie faz silnika	13
3.1.1	Wymagania wobec układu mostka tranzystorowego	13
3.1.2	Wybór odpowiednich tranzystorów	14
3.2	Zróżnice strat ciepłych na tranzystorach mocy	18
3.3	Szacowanie strat ciepłych na tranzystorach mocy	21
3.3.1	Układ symulacyjny	21
3.3.2	Straty ciepła na górnym tranzystorze	23
3.3.3	Straty dolnym na tranzystorze	24
3.4	Układy zasilania bramek	26
3.4.1	Konstrukcja układów sterowania bramek tranzystorów górnych	27
3.4.2	Konstrukcja układów sterowania bramek tranzystorów dolnych	33
3.5	Układy pomocnicze	37
3.5.1	Generowanie podwyższonego napięcia	37
3.5.2	Generowania obniżonego napięcia	38
3.5.3	Dzielniki rezystorowe oraz sztuczny punkt neutralny	39
3.6	Filtracja zasilania	39
3.7	Płytki PCB	43
3.7.1	Wybrane zagadnienia dotyczące projektowania PCB	43
3.7.2	Przedstawienie wykonanej PCB	44
4	Program sterujący	47
4.1	Użyty mikrokontroler	47
4.2	Użyte oprogramowanie	48
4.2.1	TRUEStudio	48
4.2.2	STM32CubeMX	48
4.3	Ogólna idea kodu	48
4.4	Najważniejsze elementy sprzętowe służące do realizacji programu	49
4.4.1	Układy czasowe	49

4.4.2	Przetworniki analogowo-cyfrowe i cyfrowo-analogowe	50
4.5	Najważniejsze funkcje realizujące zadanie komutacji silnika	51
4.5.1	Włączanie i wyłączanie kroków sekwencji	51
4.5.2	Komutacja oparta na opóźnieniach programowych	52
4.5.3	Komutacja z wykorzystaniem układu czasowego	52
4.5.4	Wykrycie przejścia przez zero	54
4.6	Zmiana wypełnienia sygnału PWM	55
4.6.1	Wywołanie konwersji ADC	56
4.6.2	Rozruch	57
4.7	Pozostała funkcjonalność sterownika	57
4.7.1	Odbiór sygnału PWM	57
4.7.2	Algorytm regulacji prędkości obrotowej	59
4.7.3	Filtr cyfrowy	60
4.7.4	Komunikacja z użyciem magistrali CAN	61
5	Testy i wnioski	63
5.1	Układ elektryczny	63
5.2	Filtry	64
5.2.1	Filtr cyfrowy	64
5.2.2	Filtr RC	66
5.3	Skuteczność komutacji	68
5.3.1	Komutacja na podstawie wykrywania przejść przez zero . . .	68
5.3.2	Rozruch silnika	71
5.4	Regulator prędkości obrotowej	72
5.4.1	Regulator proporcjonalny	72
5.4.2	Regulator proporcjonalny i całkujący	73
5.5	Komunikacja CAN	78
5.6	Podsumowanie wykonania celów pracy	79
5.6.1	Cele pracy	79
5.6.2	Założenia projektowe	79
	Spis ilustracji	83
	Literatura	84
	Zawartość CD	86

1 Wstęp

1.1 Cel i zakres pracy

Celem pracy jest zaprojektowanie kompletnego układu energoelektronicznego pozwalającego na zasilanie i sterowanie prędkością silnika BLDC. W części sprzętowej należało zaprojektować i wykonać część dużej mocy, bezpośrednio zasilającą silnik, oraz układy pomocnicze, a następnie sprawdzić poprawność ich działania. Część programowa polegała na zaprogramowaniu otrzymanego mikrokontrolera, tak by po połączeniu z częścią mocy, zapewniał on komutację silnika bez użycia czujników zewnętrznych oraz umożliwiał regulację prędkości obrotowej. Dodatkowo, mikrokontroler powinien odbierać sygnał PWM z odbiornika modelarskiego, oraz wysyłać informację o aktualnej prędkości obrotowej z użyciem magistrali CAN. Cały projekt wykonywany jest w kontekście robotów latających i ich zasilania. Sterownik w trakcie testów będzie więc zasilał typowy silnik używany w małych robotach latających.

Cel jest istotny, ponieważ w silniki BLDC są bardzo ważną częścią robotów mobilnych, szczególnie latających. Ich zachowanie, wynikające z ich sterowania, bezpośrednio determinuje skuteczność platform latających. Praca ta wprowadza do regulatora prędkości 32-bitowy mikrokontroler z rodziny STM32F4, który jest znacznie bardziej wydajny niż stosowane często w podobnych, komercyjnych konstrukcjach, mikrokontrolery AVR. Otwiera to drogę do zupełnie nowej funkcjonalności tych układów – umożliwia na przykład, oprócz podstawowych zadań sterowania, równoczesną obróbkę i analizę mierzonych wartości. Jako pewne pokazanie tych możliwości, w projekcie przetestowano możliwości cyfrowej filtracji sygnału z faz silnika.

1.2 Założenia projektowe

W uzgodnieniu z promotorem, poczyniono pewne założenia co do konstrukcji urządzenia:

- sterownik powinien móc zasilac silnik prądem o natężeniu 30 A po stronie prądu stałego
- układ elektroniczny nie powinien używać wyspecjalizowanych układów scalonych sterujących tranzystorami mocy
- układy inne niż tranzystory mocy, powinny móc pracować z możliwie wysokim napięciem zasilania (do ok. 30 V), tak by układ był uniwersalny.
- w czasie testowania powinna zostać sprawdzona przydatność różnych typów filtrów w do eliminacji wysokich częstotliwości z sygnału z faz silnika.

Dodatkowo, ze względów praktycznych, zdecydowano się na oddzielenie części mocy i części logicznej regulatora. W trakcie przygotowań do wykonania pracy otrzymano mikrokontroler osadzony na gotowej płytce PCB, wykonanej przez pracowników politechniki. By ograniczyć i tak już rozległy charakter projektu, użyty został gotowy układ z mikrokontrolerem, który będzie łączył się przewodami z częścią mocy.

1.3 Struktura pracy

W drugim rozdziale została opisana ogólna teoria dotycząca zarówno samego silnika BLDC, jak i ta dotycząca sterowania elektroniczną komutacją. W rozdziale tym tłumaczone są pojęcia i techniki używane dalej w pracy, szczególnie w rozdziale dotyczącym programu sterownika.

Trzeci rozdział podejmuje tematykę stworzenia układu energoelektronicznego, zarówno ze strony teoretycznej jak i praktycznej. Mówi o wyborze i konfiguracji głównych tranzystorów, budowie układów sterowania nimi, oraz o wszelkich układach pomocniczych znajdujących się na płytce. Większość omawianych tam kwestii została zbadana symulacyjnie. Na końcu rozdziału prezentowana jest też zaprojektowana płytka PCB, mieszcząca wszystkie opisane wcześniej układy.

Kolejny, czwarty rozdział, opisuje zagadnienia związane z oprogramowaniem sterownika. Traktuje o oprogramowaniu komputera, sprzętowych elementach mikrokontrolera, umożliwiających realizację komutacji, ich konfiguracji, oraz algorytmach.

Ostatni rozdział pokazuje, jak w testach praktycznych działały poszczególne elementy całego projektu i podaje wnioski z przeprowadzonych testów. Podsumowuje też realizację założeń i celów całości pracy.

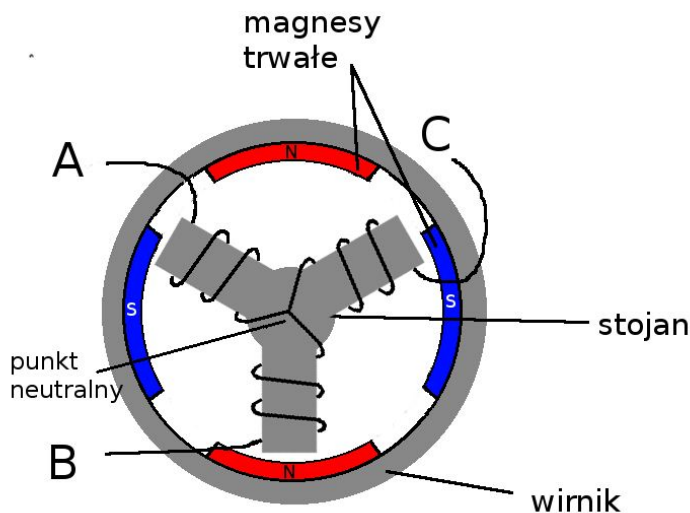
2 Teoria dotycząca sterowania silnikiem BLDC

W poniższym rozdziale opisana zostanie budowa silnika i podstawowe koncepcje związane z zasilaniem go i sterowaniem.

2.1 Budowa bezszczotkowego silnika prądu stałego

Podstawową cechą silnika jest, zgodnie z jego nazwą, brak szczotek doprowadzających zasilanie do uzwojeń. Komutacja realizowana jest z użyciem osobnego układu energoelektronicznego. Nazwa silnika może być myląca, ponieważ konstrukcja ta ma niewiele wspólnego ze szczotkowym silnikiem prądu stałego. BLDC należy do silników synchronicznych prądu zmiennego. Jediną różnicą pomiędzy nim a silnikiem synchronicznym z magnesami stałymi jest nieco inne nawinięcie uzwojeń i rozmieszczenie magnesów.[15]

Silnik składa się z wirnika z magnesami stałymi oraz statora z nawiniętymi cewkami. Zamiana energii elektrycznej na mechaniczną odbywa się przez siły magnetyczne występujące pomiędzy magnesami stałymi wirnika a wirującym polem magnetycznym indukowany przez uzwojenia statora. Większość silników tego typu ma trójfazowe uzwojenie połączone w gwiazdę. Silnika o właśnie takich uzwojeniach będą dotyczyły poniższe rozważania.



Rysunek 1: Uproszczona budowa silnika BLDC

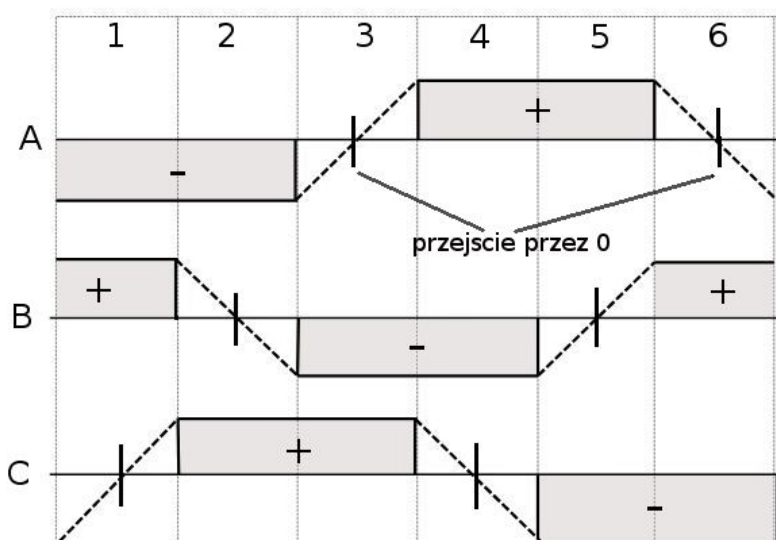
Przedstawiony na rysunku silnik został uproszczony. Rzeczywiste konstrukcje mają często więcej cewek, a jedna faza może być nawinięta na kilka cewek stojana. To samo dotyczy liczby magnesów na obwodzie wirnika.

W silnikach tych, prędkość obrotowa jest proporcjonalna do przyłożonego napięcia. Dla modelarskich silników podaje się zawsze stałą przybliżającą ten związek.

Dla silnika, z którym układ będzie testowany, wynosi ona $1400 K_V$, co należy rozumieć jako $1400 \frac{\text{obr}/\text{min}}{\text{V}}$. Tak więc silnik bez obciążenia, na którego zaciski zostanie podane napięcie 10 V i zapewniona zostanie prawidłowa komutacja, powinny obracać się z prędkością 14000 obr/min.

2.2 Schemat zasilania faz

Najważniejszym zadaniem sterownika jest wygenerowanie odpowiednich przebiegów na poszczególnych fazach silnika. Uzyskuje się je, poprzez podzielenie pełnego obrotu elektrycznego (jednego okresu przebiegu) na 6 równych kroków. Podczas każdego z kroków uzwojenia będą zasilane w inny sposób.



Rysunek 2: Uprozczone przebiegi faz silnika, rysowane względem punktu neutralnego silnika

Na powyższym rysunku pokazane zostały uproszczone przebiegi dla trzech faz silnika. Jak widać, występują tu trzy przemienne, trapezoidalne przebiegi, przesunięte względem siebie o 120 stopni (jeden krok odpowiada 60 stopniom). Zostały one narysowane względem punktu środkowego silnika. Wykres dla fazy A można zinterpretować następująco:

- w pierwszych dwóch krokach napięcie względem punktu neutralnego jest ujemne, to znaczy że zacisk A jest podłączony do masy układu.
- Następnie, w kroku 3 występuje linia przerywana, przechodząca z wartości ujemnej na dodatnią. W tym czasie zacisk A nie jest podłączony ani do masy, ani do zasilania układu, a przerywana linia oznacza napięcie indukowane w

niepodłączonej fazie. Zaznaczony pionową kreską punkt to przejście indukowanego sygnału przez 0, czyli przez napięcie punktu neutralnego.

- W kroku 4 i 5 napięcie jest dodatnie, a więc zacisk A jest podłączony do napięcia zasilania układu.
- Następnie, w kroku 6 powtarza się przejście z kroku 3, tym razem jednak sygnał przechodzi z dodatniego na ujemny.

Podobnie sytuacja ma się w przebiegach dla zacisku A i B. Korzystając z interpretacji powyższego rysunku, można wyznaczyć sekwencje zasilania dla wszystkich kroków. Tak więc w pierwszym kroku faza A będzie podłączona do masy układu, faza B do zasilania, a faza C zostanie odłączona. Podobnie czynimy ze wszystkimi krokami. Po osiągnięciu kroku 6, sekwencja zacznie się powtarzać, tworząc w ten sposób pełen obrót elektryczny.

Napięcie w wolnej fazie powstaje bez ingerencji sterownika. Jest odpowiedzią generowaną przez sam silnik, mówiącą gdzie aktualnie znajduje się wirnik. Wystąpienie przejścia przez zero oznacza, że w naturalnym przebiegu komutacji silnika, za pół okresu trwania kroku, powinna nastąpić kolejna komutacja. Informacja ta powinna zostać wykorzystana przez algorytm bezczujnikowej komutacji i to na jej podstawie powinien być ustalany moment włączenia kolejnego kroku.

Należy pamiętać, że jeden obrót elektryczny nie jest równoznaczny z mechanicznym obrotem wirnika silnika. Stosunek pomiędzy nimi określa ilość par magnesów na obwodzie wirnika. W używanym do testów silniku jest ich 14, a więc jeden obwód mechaniczny jest równy siedmiu obrotom elektrycznym. [15]

2.3 Ograniczenie natężenia przepływającego prądu

Proste podłączenie uzwojeń silnika do zasilania lub masy poprzez otwarty tranzystor, w praktyce oznaczałoby zasilanie silnika, przez cały czas, maksymalnym dostępnym napięciem zasilania. Oznaczałoby to, że pracujący już silnik dążyłby do osiągnięcia maksymalnych, możliwych dla jego konstrukcji obrotów. Natomiast przy rozruchu, gdy siła elektromotoryczna przeciwdziałająca przepływowi prądu byłaby jeszcze niewielka, przez uzwojenia płynąłby bardzo wysoki prąd, zdolny uszkodzić sterownik, ogniwa go zasilające, czy urządzenia podłączone do wału silnika (np. przekładnia).

W celu ograniczenia natężenia prądu przy rozruchu, ale także by możliwa była regulacja prędkości silnika, musi istnieć sposób ograniczenia napięcia przykładanego na jego zaciski. Najprostszym i stosowanym prawie wyłącznie rozwiązaniem (występowało ono we wszystkich sprawdzonych źródłach literaturowych), jest kluczkowanie prądu fazy poprzez wielokrotne włączanie i wyłączanie jednego z tranzystorów zasilających silnik. Najczęściej jest to wykonywane poprzez podawanie na jeden z tran-

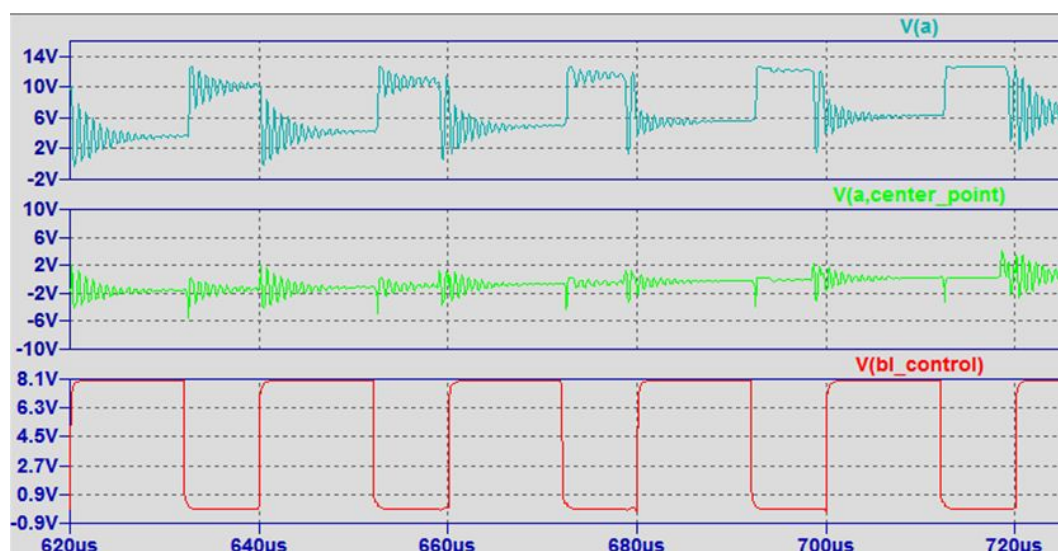
zysatorów sygnału PWM o stałym okresie i zmiennym wypełnieniu. W ten sposób, przy wypełnieniu sygnału równym 50%, tranzystor kluczujący włączony jest średnio przez połowę okresu, co dla silnika jest zbliżone do sytuacji, gdyby na jego zaciski podano napięcie równe połowie napięcia zasilania. Założenie, że sygnał PWM może zastąpić obniżenie napięcia zasilania jest prawdziwe tylko wtedy, kiedy prąd kluczowany jest z wystarczająco dużą częstotliwością. Znalezione w literaturze wartości zawierają się w zakresie 10-20 kHz, jednak można używać wyższych częstotliwości, pod warunkiem odpowiedniej konstrukcji układu energoelektronicznego.

2.4 Sposób wykrycia przejścia przez zero

Rysunek 2 zawierał napięcie narysowane względem punktu neutralnego silnika. Jego względem łatwo można wykryć przejście przez 0. Silniki modelarskie nie posiadają jednak zazwyczaj czwartego wyprowadzenia, które dawałoby dostęp do punktu neutralnego silnika. Problem ten można rozwiązać na kilka sposobów:

- zakładając, że fazy silnika są identyczne, można dokonać pomiaru na wszystkich trzech fazach jednocześnie, następnie wyliczyć średnią z dwóch zasilanych zacisków. Wyliczone w ten sposób napięcie powinno być równe potencjałowi punktu neutralnego i można je porównywać z pomiarem trzeciej, wolnej fazy, tak by wykryć przejście przez 0.
- kolejny sposób jest dalszym uproszczeniem powyższego rozwiązania. Zakładając, że jedna z faz jest bezpośrednio połączona z masą układu, a druga z zasilaniem, za napięcie punktu neutralnego można uznać połowę napięcia zasilania
- można również stworzyć sztuczny punkt neutralny, łącząc 3 identyczne rezystory na kształt uzwojeń silnika. Uznaje się wtedy, że napięcie w punkcie środkowym układu rezystorów jest równe napięciu punktu neutralnego silnika

Zdecydowano się zastosować trzeci sposób, czyli stworzenie sztucznego punktu neutralnego z rezystorów. Ma on istotne zalety w porównaniu z poprzednimi, ponieważ umożliwia on wykonanie porównań napięcia wolnej fazy i punktu środkowego przez cały okres komutacji, a także wymaga jedynie dwóch pomiarów napięcia. Pierwszy sposób wymaga wykonania za każdym razem trzech pomiarów i zwiększa ilość obrabianych danych. Drugi sposób działa tylko gdy jedna z faz jest podłączona do zasilania, druga do uziemienia, co ma miejsce tylko gdy otworzony jest tranzystor kluczujący. Spostrzeżenie te zostały potwierdzone w symulacji. Pomimo że w układzie, w fazie A, występuje przejście przez zero, potencjał pomiędzy fazą a uziemieniem ($V(a)$) nie nadaje się do jego wykrycia, w czasie gdy tranzystor kluczujący jest wyłączony.



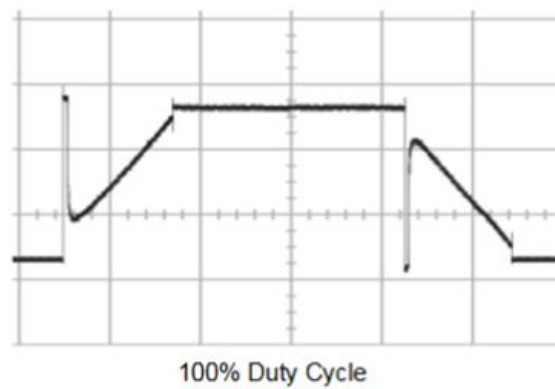
Rysunek 3: Odpowiednio: przebiegi napięcia fazy A względem uziemienia, napięcie fazy A względem punktu neutralnego, sygnał sterujący tranzystorem kluczującym prąd

Dodatkowo, na wynikach symulacji możemy zauważyć oscylacje napięcia, pojawiające się w fazie po włączeniu oraz po wyłączeniu tranzystora kluczującego. Należy się spodziewać, że w tych momentach będą się pojawiać zakłócenia, utrudniające wykrycie przejścia przez 0. Dodatkowo, zakłócenia mogą się pojawiać w samym silniku, wskutek pasożytniczych pojemności występujących pomiędzy uzwojeniami, czego symulacja nie uwzględnia.

By zminimalizować wpływ oscylacji napięcia w fazie na komutację, stosuje się różne sposoby. Jednym z nich jest filtracja napięć przed wejściem do mikrokontrolera z użyciem filtrów analogowych RC. Zostały one wykorzystane w pracy inżynierskiej na ten sam temat[9] oraz w nocie aplikacyjnej AVR444[1]. W pracy zostanie zbadana przydatność stosowania takich filtrów. Zostanie także zbadany predykcyjny filtr cyfrowy.

Innym podejściem, możliwym do zastosowania przy wykonywaniu pomiarów przetwornikiem analogowo-cyfrowym mikrokontrolera, jest wywoływanie konwersji w odpowiednim momencie. Jak widać na wykresach, oscylacje pojawiają się blisko zmian stanu tranzystora kluczującego. W tym celu, po każdym takim zdarzeniu należy odczekać kilka mikrosekund i wtedy dokonywać pomiaru. Jest to preferowany w tej pracy sposób wykonywania pomiarów napięcia, ponieważ nie wymaga on stosowania żadnych dodatkowych elementów sprzętowych.

Dodatkowo, przejścia przez zero nie należy wykrywać na samym początku kroku, ponieważ przez fazę może w dalszym ciągu płynąć prąd, będący pozostałością po poprzednim kroku komutacji.



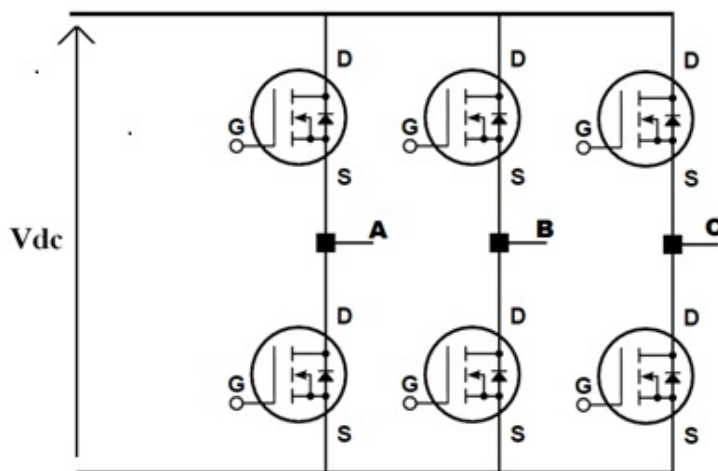
Rysunek 4: Napięcie fazy zasilanego silnika przy wypełnieniu PWM równym 100%. Pokazuje skoki napięcia pojawiające się przy zmianie kroku ¹.

¹Pochodzi z [2]*Brushless DC Motor Control Made Easy*

3 Główny układ energoelektroniczny

3.1 Zasilanie faz silnika

Do prawidłowej pracy silnika niezbędna jest możliwość wygenerowania przebiegów jak rysunku 2. Konieczne jest zapewnienie możliwości podłączenia każdej z trzech faz silnika do napięcia zasilania lub masy. Powszechnie używanym do zasilania silników synchronicznych układem tranzystorów dającym takie możliwości jest mostek H, rozszerzony o dodatkową gałąź, potrzebną ze względu na obecność trzeciej fazy.



Rysunek 5: Typowa konfiguracja trójfazowego mostka H

3.1.1 Wymagania wobec układu mostka tranzystorowego

Podstawowymi warunkami, jakie muszą spełnić tranzystory użyte w układzie są:

- możliwość przewodzenia odpowiedniego prądu
- możliwość włączania i wyłączania klucza z odpowiednią częstotliwością

Sterownik powinien być obciążalny prądem o natężeniu 30 A płynącym w szynie prądu stałego. Biorąc pod uwagę, że w dowolnej chwili aktywnie przewodzą prąd tylko dwa z sześciu tranzystorów, wartość skuteczna natężenia prądu, w całym okresie, dla pojedynczego elementu będzie niższa niż 30 A. Chwilowe wartości natężenia będą przekraczać 30 A. Jak mocno - zależy to od parametrów (głównie indukcyjności) podłączonego silnika.

Podstawową częstotliwość przełączania się wszystkich tranzystorów wyznacza częstotliwość zmiany kroków. Ponieważ mamy do czynienia z silnikiem synchronicznym, ściśle zależy ona od aktualnej szybkości obrotowej.

W celu regulacji mocy silnika z wykorzystaniem kluczowania przepływającego

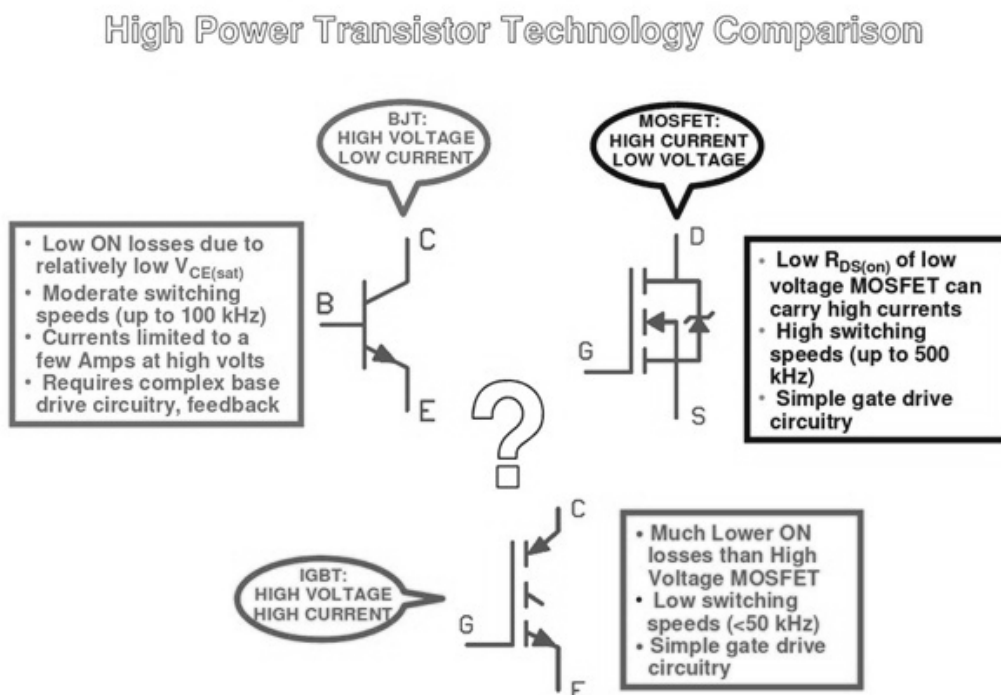
prądu, jeden z aktywnych w dowolnym momencie tranzystorów będzie musiał przełączać się z częstotliwością przynajmniej dziesięciokrotnie większą niż częstotliwość komutacji silnika. By zapewnić większą elastyczność sterownika, tranzystory powinny mieć możliwość skutecznego kluczenia prądu z częstotliwością większą niż 20 kHz. Zwiększenie częstotliwości będzie konieczne na przykład w przypadku pracy z wyższym napięciem zasilania, silnikiem o wyższej stałej K_V , lub potrzebą wyjścia poza zakres słyszalny. W praktyce stosuje się minimalne częstotliwości zapewniające skuteczną regulację prądu, właśnie z przedziału 8-20 kHz.

3.1.2 Wybór odpowiednich tranzystorów

Przy budowie układu energoelektronicznego możemy rozważać 3 podstawowe typy tranzystorów:

- tranzystory bipolarne (BJT)
- tranzystory bipolarne z izolowaną bramką (IGBT)
- tranzystory polowe (FET), głównie tranzystory o strukturze metal, tlenek, półprzewodnik (MOSFET)

Dla ułatwienia wyboru, stosuje się często upraszczające podziały określające pole zastosowania elementu:



Rysunek 6: porównanie technologii tranzystorów wysokiej mocy. Opis: BJT: wysokie napięcie, niski prąd, MOSFET: wysoki prąd, niskie napięcie, IGBT: wysokie napięcie, wysoki prąd ²

Ze względu na warunki pracy mostka (natężenie prądu do około 50 A, napięcie mniejsze niż 30V) nasza uwaga powinna się skupić na tranzystorach typu MOSFET. Jednym z podstawowych parametrów, pozwalającym ocenić przydatność typu tranzystora do danego zastosowania jest oszacowanie strat cieplnych w elemencie w trakcie pracy. Ciepło wydzielane w tranzystorze zależy od wydzielanej w nim mocy czynnej, czyli jest funkcją spadku napięcia i natężenia przepływającego prądu. Dla otwartego tranzystora typu MOSFET, spadek napięcia jest zależny od rezystancji przewodzenia ($R_{DS(on)}$) i natężenia przepływającego prądu. Ogólne straty cieplne opisuje więc w przybliżeniu wzór $I^2 R_{dson}$. Natomiast dla tranzystorów bipolarnych, spadek napięcia (V_{cesat}) jest wartością w przybliżeniu stałą w pewnym zakresie natężeń prądu, rosnącą jednak bardzo szybko po opuszczeniu tego zakresu.

Znając powyższe fakty oraz parametry typowych tranzystorów nadających się do zastosowania w sterowniku silnika, możemy przystąpić do szacowania strat cieplnych.

Dla tranzystora MOSFET:

²źródło: [6] FET vs. BJT vs. IGBT: What's the Right Choice for Your Power Stage Design?

R_{DSon} - zazwyczaj $1 - 50m\Omega$. Za wartość typową można uznać $10m\Omega$

$I = 20A$

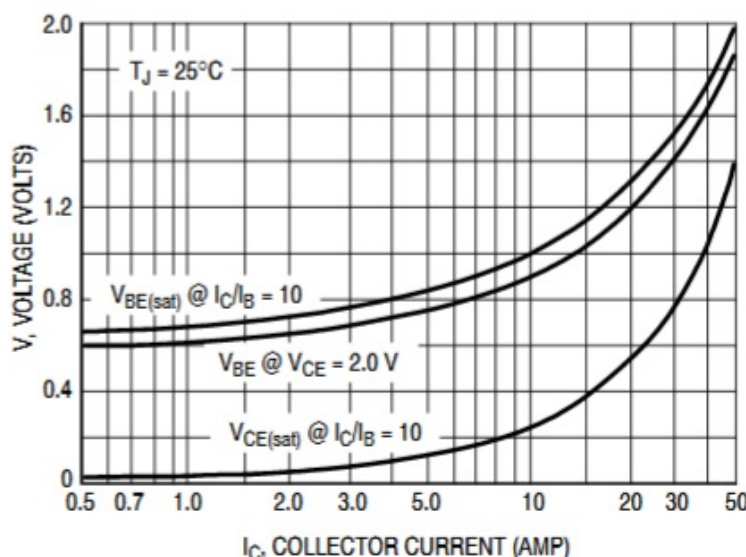
$$P = I_D^2 R_{DSon} = 20^2 \cdot 0.01 = 4W \quad (1)$$

Dla tranzystora bipolarnego:

V_{CEsat} = w zakresie 0.15-2V. Dla przykładowego tranzystora 2N5684 będzie wynosić około 0.6V (rys. 7)

$$P = UI = 0.6 \cdot 20 = 12W \quad (2)$$

W wybranym tranzystorze, do strat dodatkowo należy doliczyć moc wydzielaną jako skutek przepływu prądu przez bazę tranzystora, około 1.5W (ponieważ prąd bazy wynosiłby około 2 A).



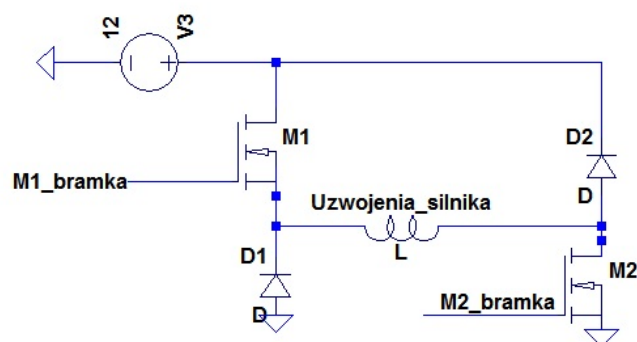
Rysunek 7: Zależność napięcia nasycenia od prądu kolektora³

Jak widać, w wybranym przykładzie znaczną przewagę ma tranzystor typu MOSFET. Szukając odpowiednich elementów, można się przekonać, że tranzystory bipolarne zdolne przewodzić prądy do 50 A są dosyć rzadkie, a dodatkowo występują głównie w bardzo dużych obudowach przewlekanych. Tymczasem tranzystory MOSFET o $R_{DSon} = 5m\Omega$ (w powyższym przykładzie byłoby to jednoznaczne z wydzielaniem 2 W ciepła) są dostępne w małych, montowanych powierzchniowo obudowach, a ich dostępność jest bardzo wysoka.

Tranzystory MOSFET dzieli się ze względu na rodzaj kanału – w sterownikach silników stosuje się jednak prawie wyłącznie te z kanałem wzbogacanym. Są one normalnie wyłączone, a przewodzący kanał tworzy się dopiero, gdy napięcie pomiędzy bramką(G) a źródłem(S) przekroczy pewną wartość progową. MOSFETy z kanałem

³Obraz pochodzi z karty katalogowej tranzystora 2N5684, ON Semiconductors

wzbogacającym można dodatkowo podzielić na te z kanałem typu N i typu P. Podział ten jest istotny, ponieważ oba te typy tranzystorów stosowane są w sterownikach silników BLDC, różnią się one natomiast co do swoich właściwości i sposobu sterowania. W tranzystorach z kanałem typu N, w celu otwarcia kanału, należy na bramkę podać napięcie dodatnie z stosunku do źródła, a przypadku kanału typu P – ujemne. W naszym układzie występują tranzystory znajdujące się pomiędzy dodatnim napięciem a obciążeniem (M1 na rys. 4) oraz znajdujące się pomiędzy obciążeniem a masą (M2), które dalej będą nazywane odpowiednio tranzystorami górnymi i dolnymi.



Rysunek 8: Uproszczony schemat zasilania silnika

Ujemne napięcie bramki sprawia, że tranzystorem typu P można łatwiej sterować, gdy znajduje się on powyżej obciążenia, ponieważ z celu uzyskania ujemnego napięcia GS, wystarczy połączyć bramkę tranzystora z masą. W przypadku dolnego tranzystora, korzystniejszy jest typ N, ponieważ do otwarcia kanału wystarczy przyłożyć na jego bramkę dodatnie napięcie.

Jednak nie jest to jedyna właściwość odróżniająca oba typy. Dla MOSFETów typu N, głównym nośnikiem ładunku są elektrony, które są bardziej ruchliwe od przenoszących ładunek dziur, występujących w przypadku typu P. W związku z tym, typ N charakteryzuje się niższą rezystancją przewodzenia niż typ P, zbudowany na tej samej wielkości matrycy krzemowej. Sprawia to, że tranzystory z kanałem typu N lepiej nadają się do układów przewodzących prądy o dużym natężeniu.[7]

Projektant musi więc rozważyć, czy postawić nacisk na prostszy układ sterowania, czy mniejsze straty cieplne na elemencie. W opisywanym projekcie, ze względu na wysokie wymagania dotyczące przewodzonego natężenia prądu, zdecydowano się na użycie tranzystorów z kanałem typu N, kosztem bardziej skomplikowanych układów zasilających bramkę.

Głównie ze względu na dobrą dostępność i niską cenę, wybrano tranzystor IRLR8743, firmy International Rectifier. Zgodnie z kartą katalogową (zdjęcie niżej), jest on przeznaczony do wysokoczęstotliwościowych, niskonapięciowych przetwornic oraz cha-

rakteryzuje się bardzo niską rezystancją przewodzenia. Cechy te są korzystne dla omawianego tutaj układu, ponieważ umożliwią zredukowanie strat cieplnych.

International
IR Rectifier

PD - 96123

IRLR8743PbF
IRLU8743PbF

HEXFET® Power MOSFET


Applications

- High Frequency Synchronous Buck Converters for Computer Processor Power
- High Frequency Isolated DC-DC Converters with Synchronous Rectification for Telecom and Industrial Use
- Lead-Free

Benefits

- Very Low $R_{DS(on)}$ at 4.5V V_{GS}
- Ultra-Low Gate Impedance
- Fully Characterized Avalanche Voltage and Current

V_{DSS}	$R_{DS(on)}$ max	Qg
30V	3.1m Ω	39nC



G	D	S
Gate	Drain	Source

Absolute Maximum Ratings

	Parameter	Max.	Units
V_{DS}	Drain-to-Source Voltage	30	V
V_{GS}	Gate-to-Source Voltage	± 20	
I_D @ $T_C = 25^\circ\text{C}$	Continuous Drain Current, $V_{GS} @ 10\text{V}$	160 ^①	A
I_D @ $T_C = 100^\circ\text{C}$	Continuous Drain Current, $V_{GS} @ 10\text{V}$	113 ^①	
I_{DM}	Pulsed Drain Current ^①	640	
P_D @ $T_C = 25^\circ\text{C}$	Maximum Power Dissipation ^②	135	W
P_D @ $T_C = 100^\circ\text{C}$	Maximum Power Dissipation ^②	68	

Rysunek 9: Część karty katalogowej IRLR8743

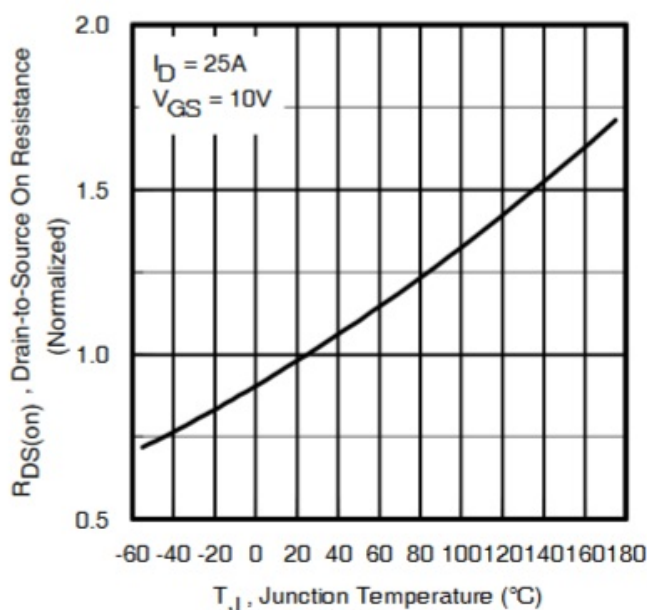
3.2 Źródła strat cieplnych na tranzystorach mocy

Jako że straty cieplne na elementach są z jednym z podstawowych kryteriów wpływających na poprawność pracy układu, będą miały wpływ nie tylko na wybór typu tranzystora (jak w poprzednim rozdziale), ale także na konstrukcję układów zasilających tranzystory mocy, zdecydowano się rozpatrzeć je nieco obszerniej.

Podstawowymi stratami w tranzystorach MOSFET są:

- straty w stanie przewodzenia
wynikają z rezystancji kanału otwartego tranzystora ($R_{ds(on)}$). Przy zmienia-
niu napięcia na bramce, w okolicach napięcia progowego ($V_{GS(th)}$), wartość ta
szybko spada. Przy dalszym zwiększaniu napięcia, trend ten wyhamowuje, a
dodatkowe zwiększanie napięcia na bramce nie przynosi już wyraźnej różnicy
w $R_{ds(on)}$, zmniejsza za to żywotność tranzystora (przy $|V_{GS}| = 20\text{V}$ większość

tranzystorów tego typu może już ulec uszkodzeniu). Z karty katalogowej wybranego tranzystora wynika, że bardzo niską rezystancję drenu uzyskujemy już przy 4.5 V. Wpływ na omawiany parametr ma także temperatura złącza tranzystora (T_J). Na rysunku niżej widać, że tranzystor przegrzany ($T_J > 175^\circ\text{C}$), może mieć rezystancję złącza nawet 70% większa niż tranzystor zimny. Jest to dodatkowy argument za staraniem się o jak najmniejsze wydzielanie ciepła na tranzystorach MOSFET.



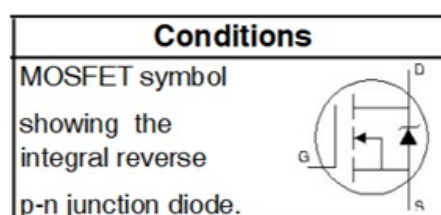
Rysunek 10: zależność pomiędzy rezystancją przewodzenia a temperaturą złącza tranzystora karty katalogowej IRLR8743

- **Straty przełączeniowe**
pojawiają się a trakcie włączania lub wyłączania tranzystora. Przez krótki czas, w fazie przełączania, gdy rezystancja drenu zmienia się z bardzo wysokiej na bardzo niską i odwrotnie, tranzystor jest podatny na wydzielanie się na nim dużych mocy. Ze względu na indukcyjną naturę przełączanego obciążenia, będzie to widoczne szczególnie przy wyłączaniu się klucza. Czas trwania tej fazy należy możliwie skrócić.
Częścią składową strat przełączeniowych są też te związane z prądem potrzebnym do ładowania pasożytniczych pojemności, występujących pomiędzy zaciskami tranzystora. Jednak ze względu na stosunkowo niskie częstotliwości występujące w układzie, ich bezpośrednie skutki cieplne są prawdopodobnie pomijalne. Ogólnie, straty przełączeniowe zwiększają się wraz ze wzrostem częstotliwość. Energia potrzebna do przeładowywania pojemności pasożyt-

niczych staje się niepomijalna dla układów pracujących z bardzo wysokimi częstotliwościami (megaherce), na przykład dla falowników rezonansowych.⁴

- Straty na wewnętrznej diodzie

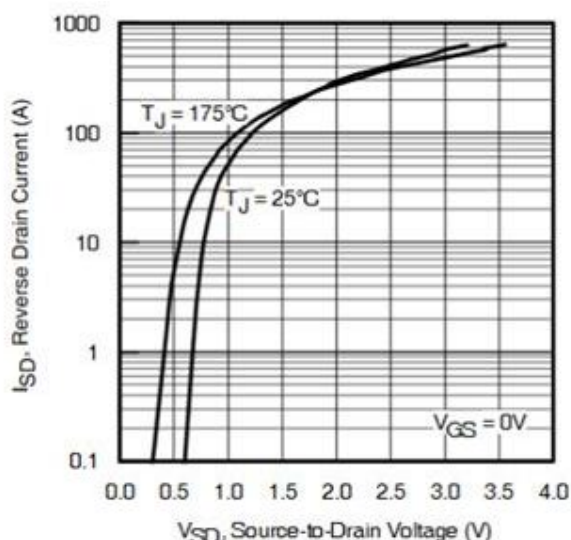
Przełączanie obciążeń indukcyjnych tranzystorami najczęściej wymaga użycia dodatkowych diod, przewodzących przeciwnie do otwartego tranzystora, by zapewnić alternatywną ścieżkę dla prądu w przypadku nagłego zamknięcia klucza. Tranzystory typu MOSFET, ze względu na swoją budowę wewnętrzną, zawierają w sobie diodę, nazywaną podłożową. Symbol tranzystora z uwzględnieniem tego faktu zawiera rys. 11



Rysunek 11: Symbol tranzystora z uwzględnieniem wewnętrznej diody z karty katalogowej IRLR8743

Dioda ta, jak już wspomniano, przewodzi gdy zamknięta zostanie normalna ścieżka przepływu prądu. Przepływ prądu przez ten element będzie skutkował spadkiem napięcia, a przez to także wydzielaniem ciepła. Zależność pomiędzy spadkiem napięcia na diodzie a prądem przepływającym ze źródła do drenu pokazuje rysunek 12.

⁴Straty mocy i rezystancja zastępcza związane z przeładowywaniem nieliniowej pojemności wyjściowej tranzystora MOSFET, Z. KACZMARCZYK, M. ZELLNER, K. FRANIA, Politechnika Śląska
<http://pe.org.pl/articles/2018/3/10.pdf>



Rysunek 12: zależność pomiędzy natężeniem prądu a spadkiem na diodzie wewnętrznej tranzystora z karty katalogowej IRLR8743

3.3 Szacowanie strat ciepłych na tranzystorach mocy

Ze względu na obecność stanów nieustalonych i skomplikowanych przebiegów związanych z włączaniem i wyłączaniem tranzystorów, do szacowania strat ciepłych wykorzystano symulację w programie LT SPICE XVII.

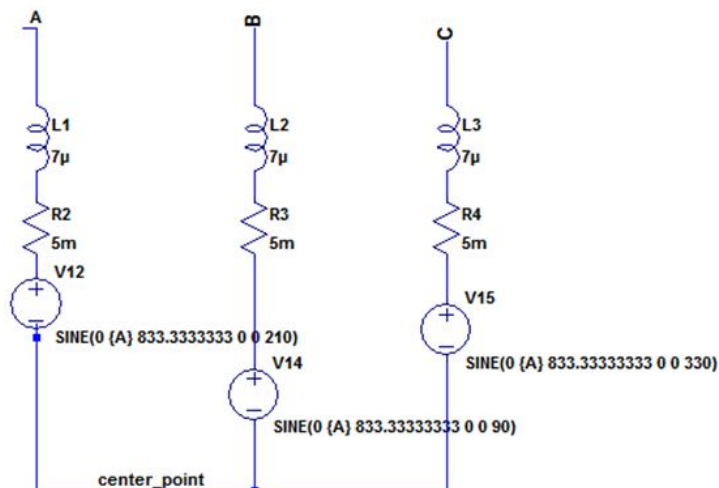
3.3.1 Układ symulacyjny

W programie LT SPICE XVII stworzono układ elektryczny, zgodny z finalną wersją fizycznego sterownika. Dodatkowo, dodano elementy mające symulować pracujący silnik BLDC.

Ustalono następujące parametry symulacji:

- napięcie zasilania – 12V
- natężenie prądu na szynie prądu stałego (źródło V3) równe 28A RMS
- częstotliwość komutacji elektrycznej – 833Hz, odpowiadająca zasilaniu silnika obracającego się z szybkością 7140 obr/min
- wypełnienie sygnału kluczującego – 92 procent, częstotliwość sygnału – 16.7kHz
- elementy dobrano tak, by w maksymalnym stopniu odpowiadały rzeczywistym komponentom, wykorzystanym do budowy układu. Jako główne tranzystory wybrano model IRFH7932, ponieważ był on dostępny w bibliotece programu. Ma on bardzo podobne parametry do IRLR8743, zarówno co do rezystancji drenu, jak ładunku bramki, wykonany jest też w tej samej technologii HEXFET.

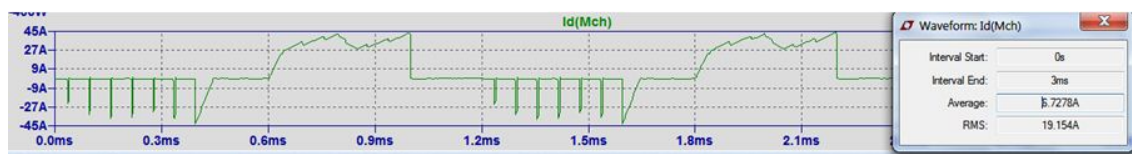




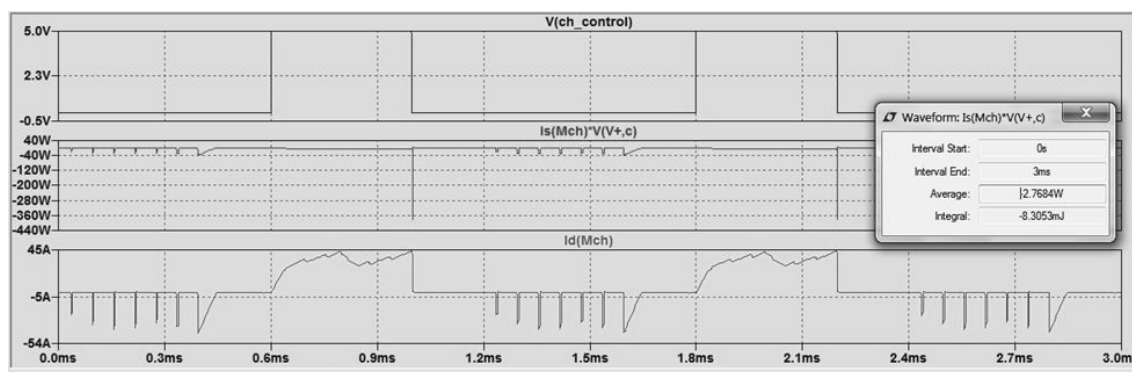
Rysunek 14: Część układu symulująca pracujący silnik

3.3.2 Straty ciepła na górnym tranzystorze

Wyniki symulacji:



Rysunek 15: Przebieg prądu drenu górnego tranzystora

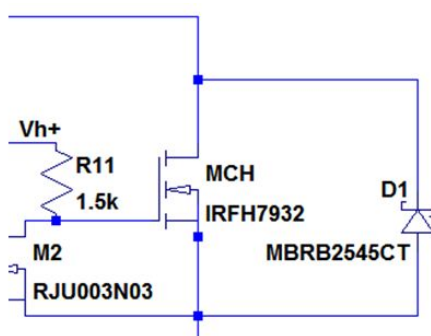


Rysunek 16: Zestawienie sygnału sterującego(górny wykres), wydzielanej mocy(środkowy wykres) oraz prądu drenu górnego tranzystora

Prąd drenu MCH wynosi 19.1 A RMS. Wydzielane ciepło, liczone jako iloczyn spadku napięcia i natężenia prądu źródła, jest równe 2.77 W. Korzystając z noty katalogowej IRLR8743, dowiemy się że przy odpowiedniej konstrukcji PCB, maksymalna rezystancja termiczna pomiędzy złączem a otoczeniem wynosi 50°C/W. Oznacza to, że jeśli tranzystor byłby zamocowany na PCB, a temperatura otoczenia

wynosiła 20 °C, wzrost temperatury złącza wynosiłby 138.5 °C, co oznacza temperaturę złącza równą 158 °C, a ta jest bliska maksymalnej temperaturze podanej w karcie katalogowej (175° C). Biorąc pod uwagę dodatkowe czynniki, jak wzrost rezystancji drenu wraz z temperaturą, czy bliskość innych tranzystorów, oznacza to, że tranzystor bez żadnego dodatkowego chłodzenia może być wystawiony na działanie temperatury przekraczającej dane katalogowe. Jednak powszechną praktyką w podobnych konstrukcjach jest dodawanie na tranzystory radiatorów. Obniżenie rezystancji termicznej pomiędzy złączem a otoczeniem do 40 °C/W zmniejszyłoby wzrost temperatury do 111 °C, co powinno być wystarczające do bezpiecznej pracy tranzystora. Dodatkowo należy pamiętać, że sterowniki pracują najczęściej w otoczeniu z bardzo szybkim przepływem powietrza.

Dodatkowo, korzystając z symulacji można określić, który rodzaj strat w tym przypadku dominuje. Dodanie równolegle dodatkowej diody, o niższym napięciu przewodzenia niż wewnętrzna dioda tranzystora, powinno spowodować przekierowanie większości prądów wstecznych na nią. Po modyfikacji układu okazuje się, że



Rysunek 17: Schemat otoczenia górnego tranzystora, z dodatkową diodą

na tranzystorze MCH wydziela się tylko 1.2 W, a natężenie prądu wynosi 17.5 A RMS. Dodatkowo, możemy sprawdzić, jaka moc powinna się wydzielać przy takim natężeniu prądu jako skutek rezystancji R_{DSon} .

Moc wydzielana z prawa Ohma:

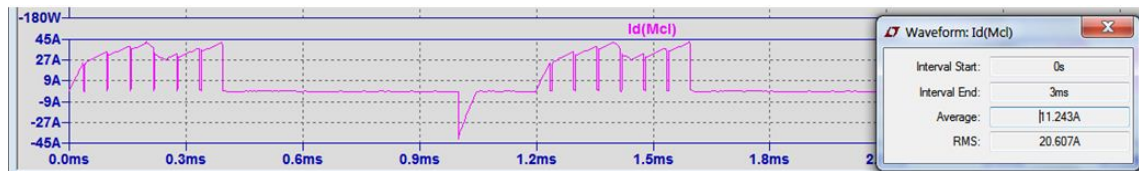
Dla zastosowanego tranzystora rezystancja przewodzenia $R = 3m\Omega$

$$I^2 R = 17.5^2 \cdot 0.003 = 0.95W$$

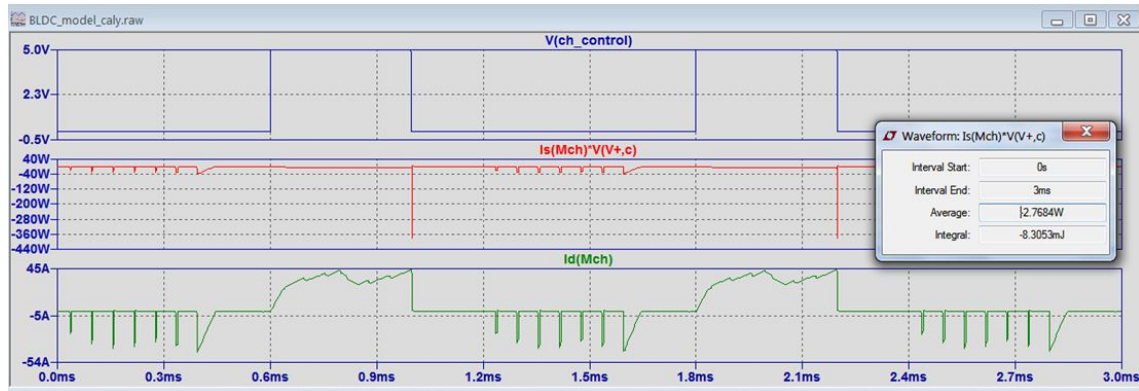
Możemy więc wywnioskować, że większość wydzielanej mocy jest skutkiem przepływu prądu przez wewnętrzną diodę. Kolejna, znacząca część pochodzi z rezystancji otwartego kanału, a pozostała moc może wynikać ze strat przełączeniowych.

3.3.3 Straty dolnym na tranzystorze

Poniżej przedstawione zostały wyniki symulacji, umożliwiające określić straty cieplne. Prąd drenu MCH jest równy 20.6 A RMS

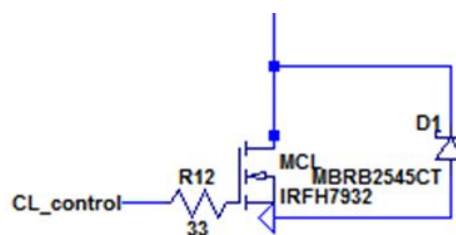


Rysunek 18: Przebieg prądu drenu dolnego tranzystora

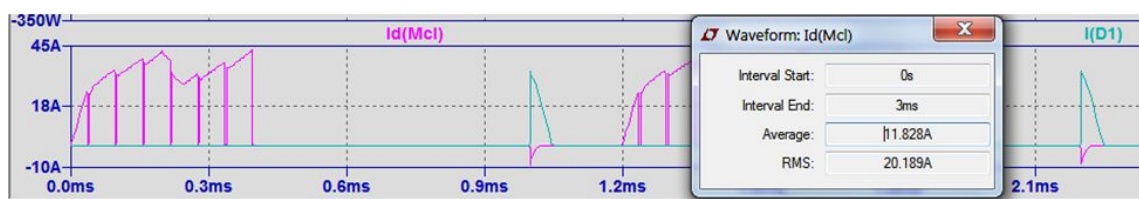


Rysunek 19: Zestawienie sygnały sterującego, wydzielanej mocy i prądu drenu dolnego tranzystora

Wydzielana moc jest równa 1.921 W. Ponieważ jest ona mniejsza niż dla tranzystorów z poprzedniego rozdziału, jeśli zarówno tranzystory górne jak i dolne będą pracować w tych samych warunkach, na tej samej PCB i na tym samym radiatorze, rozważania na temat temperatury pracy tranzystorów dolnych można pominąć. Następnie dodano diodę, podobnie jak w poprzednim punkcie. Spowodowało to zmniejszenie wydzielanej mocy do 1.43 W. Jak widać, dla tranzystora dolnego, straty pochodzące z wewnętrznej diody są znacznie mniejsze.



Rysunek 20: Schemat otoczenia dolnego tranzystora z dodatkową diodą



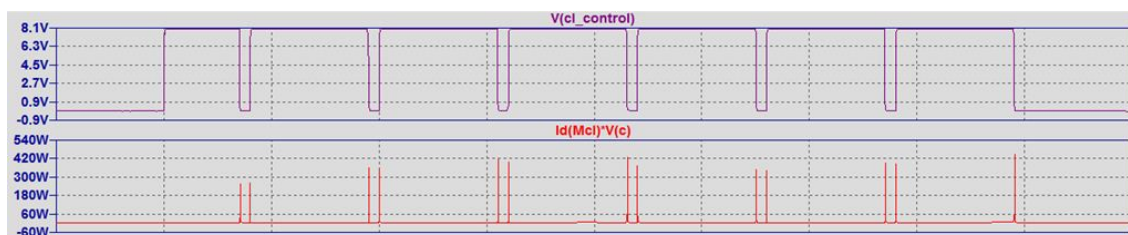
Rysunek 21: Przebieg prądu tranzystora i dodatkowej diody

Moc wydzielana z prawa Ohma:

$$\text{Dla } R_{DSon} = 3m\Omega$$

$$I^2 R = 20.2^2 \cdot 0.003 = 1.22W$$

Oznacza to też, że ponownie, straty przełączeniowe stanowią niewielką (<200 mW) część ogólnych strat na tranzystorze. Przyglądając się jednak wykresowi wydzielanej mocy, widać wyraźnie, że każde przełączenie skutkuje nagłym wzrostem wydzielania ciepła, sięgającym kilkuset watów. Należy się więc spodziewać, że mały udział tego rodzaju strat jest skutkiem doboru tranzystora, zoptymalizowanego pod kątem szybkiego przełączania oraz odpowiedniego sterowania jego bramką.



Rysunek 22: Zestawienie sygnału sterującego i ciepła wydzielanego na dolnym tranzystorze

W późniejszym przebiegu pracy, w czasie testów, okazało się, że zwiększenie częstotliwości kluczowania prądu (nawet do około 50 kHz), pozytywnie wpływa na jakość działania algorytmu bezczujnikowej komutacji. Zmieniono więc częstotliwość i ponownie sprawdzono wydzielanie ciepła na dolnym tranzystorze. Wynosiło 2.3 W, co oznacza wzrost o około 0.4 W w stosunku do częstotliwości 16.7 kHz.

3.4 Układy zasilania bramek

Kolejną, równie ważną, częścią układu energoelektronicznego są układy sterowania bramek. Każdy z układów musi przede wszystkim zapewniać skuteczną kontrolę nad stanem przewodzenia tranzystorów mocy. Musi zapewniać ich pełne otwieranie i zamykanie, w ścisłych ograniczeniach czasowych. Nieprawidłowe działanie któregoś z nich, w najlepszym przypadku, doprowadzi do złej komutacji i zatrzymania silnika, a w najgorszym – do zniszczenia całego układu. Dodatkowo, ze względu na konstrukcję sterownika, przerwanie któregoś z przewodów doprowadzających sygnały cyfrowe, musi być jednoznaczne z pewnym wyłączeniem tranzystora, którego ten sygnał dotyczy. Przy projektowaniu takiego układu należy pamiętać o pasożytniczych pojemnościach i minimalizacji strat cieplnych na sterowanych tranzystorach mocy. W tym celu, natężenia prądów dostarczanych do bramek powinny być możliwe wysokie.

3.4.1 Konstrukcja układów sterowania bramek tranzystorów górnych

Bardzo prosty układ, umożliwiający sterowanie tranzystorem górnym, został przedstawiony w nocie aplikacyjnej *AND9093/D: Using MOSFETs in LoadSwitch Applications*. [7]

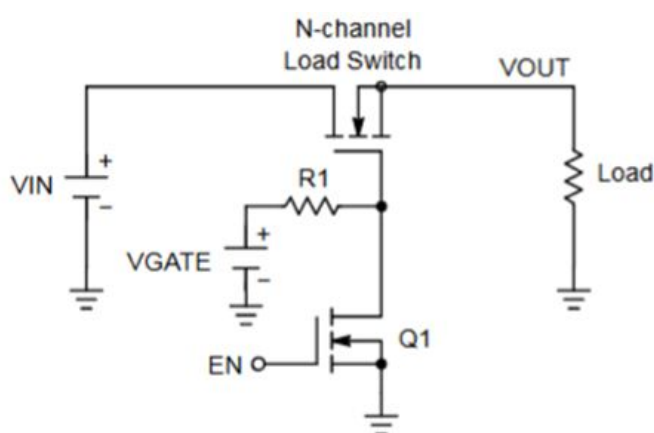
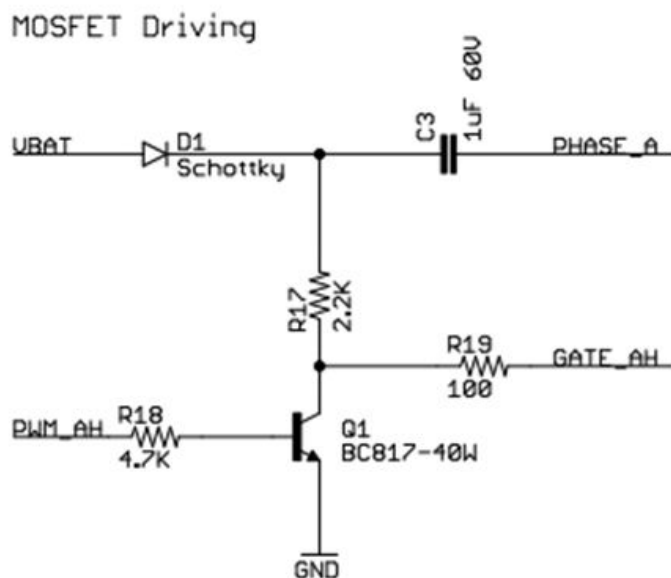


Figure 2. N-channel Example Control Circuit

Rysunek 23: Przykładowy układ sterujący bramką z AND9093/D

Schemat przedstawia sterowanie tranzystorem z kanałem typu N, z obciążeniem znajdującym się poniżej tranzystora. Z taką samą sytuacją mamy do czynienia w naszym układzie, przedstawiony układ może być więc źródłem inspiracji.

Konsekwencją wyboru MOSFETu z kanałem typu N jest fakt, że napięcie VGATE musi być wyższe od VIN. Tylko w takim przypadku, po otwarciu tranzystora (gdy $V_{IN} = V_{OUT}$), uda się utrzymać potencjał bramki wyższy niż źródła. Karta katalogowa wspomina, że nasz tranzystor ma bardzo niską rezystancję drenu przy napięciu $V_{GS} = 4.5 \text{ V}$. Napięcie VGATE powinno więc wynosić minimum $V_{IN} + 4.5 \text{ V}$. Jak więc widać, zachodzi potrzeba wygenerowania w jakiś sposób, napięcia wyższego niż napięcie zasilania. Podobny układ, zawierający jednak rozwiązanie problemu wyższego napięcia, zawiera dokumentacja sterownika BlueESC[5] firmy BlueRobotics:



Rysunek 24: Schemat układu sterowania bramką (GATE AH) sterownika BlueESC

Do generowania wyższego napięcia jest tutaj wykorzystywana dioda i kondensator, w układzie nazywanym bootstrap. Naturalnie występujące w układzie skoki napięcia fazy, są wykorzystywane do ładowania kondensatora do podwojonego napięcia zasilania. Rozwiązanie jest to stosunkowo proste, ma jednak kilka wad:

1. Kondensator C3 zbiera dodatkowy ładunek, pozwalający zwiększyć napięcie, tylko jeśli na fazie występują duże skoki napięcia. Stała czasowa układu RC utworzonego z R17 i C3 wynosi tylko 2.2 ms. Zakładając, że kondensator przy rozładowaniu przez rezystor, utrzyma użyteczne dla układu napięcie 30 procent stałej czasowej, czyli $730 \mu\text{s}$, sterownik prawdopodobnie nie mógłby pracować z wypełnieniem PWM równym 100 procent. Dla takiego przypadku, główną częstotliwością występującą na fazie jest częstotliwość komutacji, wynoszącą np. 500 Hz, co oznacza okres drgań równy 2 ms. Zwiększanie rezystancji R17 zmniejszy prąd ładujący bramkę MOSFETu, a zwiększenie pojemności kondensatora mogłoby oznaczać potrzebę zastosowania kondensatora elektrolitycznego.
2. Rezystor R17, ze względu na to, że na jego zaciskach występuje bardzo duża różnica potencjałów (nawet 40 V, przy zasilaniu układu dwudziestoma voltami), jest narażony na wydzielanie dużych mocy, szczególnie, biorąc pod uwagę, że przewodzi on prąd w sposób ciągły przez $2/3$ czasu. W takim przypadku z rezystora trzeba by odprowadzić aż 0.5 W ciepła.
3. Przy takiej konfiguracji, gdzie bramka jest naprzemiennie podciągana do podwojonego napięcia zasilania lub masy, jest ona narażona na przekroczenie

maksymalnego napięcia V_{GS} tranzystora mocy, które wynosi tylko 20 V. Taki układ nie mógłby pracować też z pewnością pracować z napięciami zasilania wyższymi od 20 V. Warto wspomnieć, że w przeciwieństwie do maksymalnego napięcia V_{DS} , którego wartość można dobierać, w zakresie do kilkuset woltów, napięcie V_{GS} dla tranzystorów MOSFET z reguły nie przekracza 20 V.

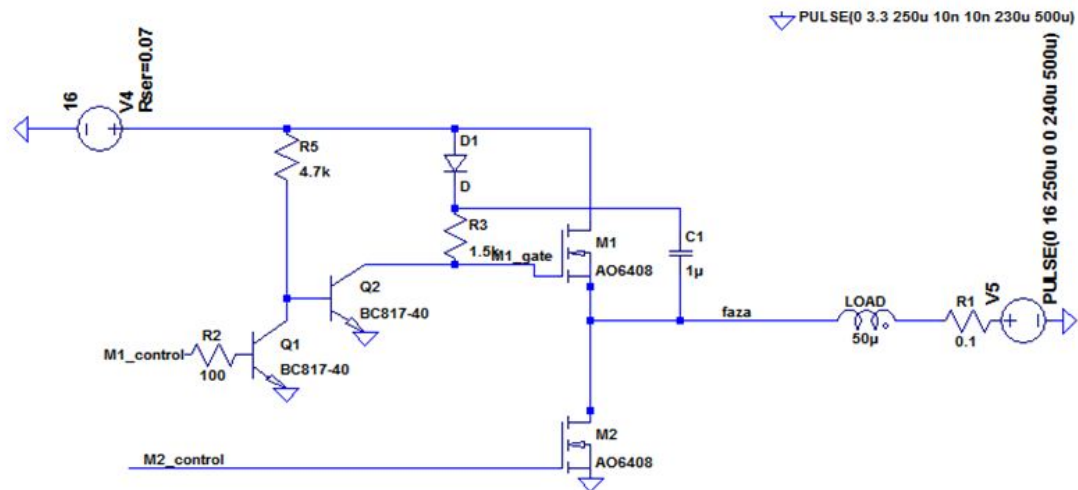
Żeby zaradzić tym problemom, zdecydowano się skonstruować własny układ. Poinstalowano jednak zachować jego podstawowe cechy. Rezystor ładujący bramkę w celu włączenia tranzystora oznacza niski prąd ładowania, a tym samym stosunkowo wolne otwieranie tranzystora górnego. Jednak nie jest to problem, ponieważ niewielkie opóźnienie we włączeniu tranzystora nie spowoduje niebezpieczeństwa zwarcia zasilania, dodatkowo, powolne włączanie tranzystora na początku kroku elektrycznego nie skutkuje nagłym wzrostem wydzielanego ciepła. Znacznie ważniejsze dla bezpieczeństwa jest możliwie szybkie wyłączenie tranzystora, co jest tutaj zapewniane przez tranzystor rozładowujący, połączony bezpośrednio z bramką.

Jako odpowiedź na wszystkie problemy powyższej konstrukcji, na płycie sterownika umieszczono samodzielny układ, generujący napięcie o 6V wyższe od napięcia zasilania. Dzięki temu możliwe będzie używanie wypełnienia PWM równego 100 procent, ponieważ napięcie zasilające bramki będzie generowane niezależnie. Odpowiednik rezystora R17 nie będzie też narażony na tak duże rozpiętości napięć na jego zaciskach, co oznacza mniejsze wydzielanie ciepła. Najbardziej istotna różnica dotyczy punktu trzeciego – dodatkowy układ, generujący napięcie o stałą wyższą niż zasilanie, otwiera drogę do możliwości zastosowania dowolnych napięć zasilania, bez obawy o zniszczenie tranzystora poprzez przekroczenie maksymalnego napięcia V_{GS} .

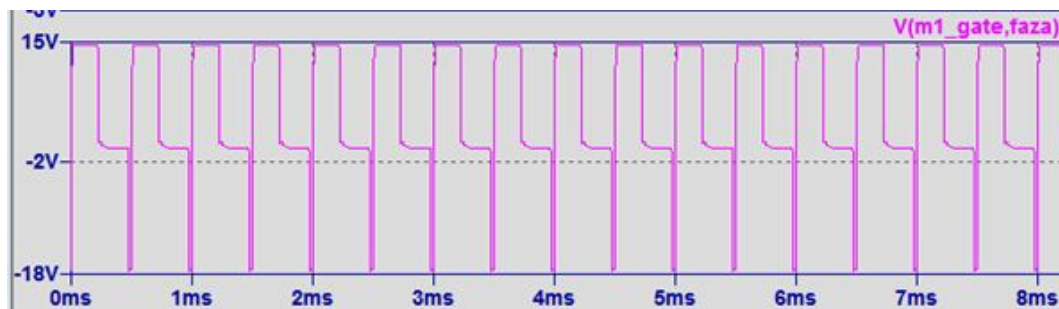
Żeby lepiej rozwiązać problem zawarty w punkcie trzecim, przeanalizujmy dokładniej przebiegi napięć V_{GS} dla różnych konfiguracji układu. W środowisku symulacyjnym stworzono uproszczony układ, będący jedną z części mostka, zawierający układ sterowania bramką, analogiczny do sterownika BlueESC (rys. 24).

Jak widać przy napięciu zasilania równym 16V, napięcie V_{GS} waha się od 15 do prawie -18V, co już wprowadza pewne niebezpieczeństwo przekroczenia +20V.

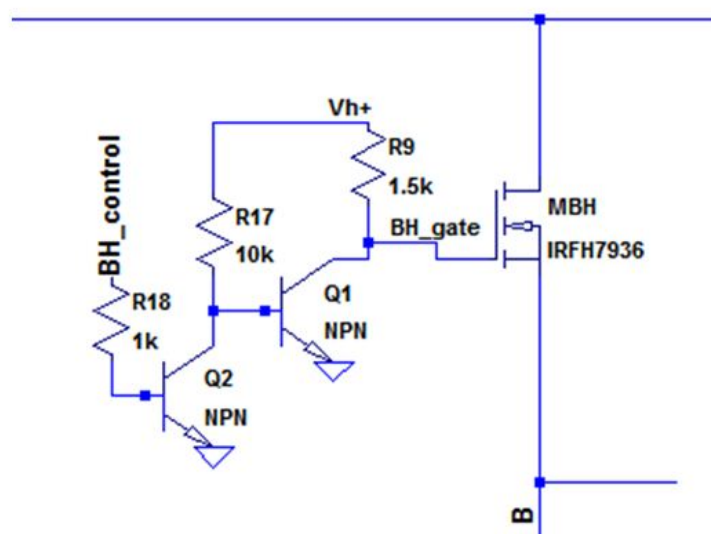
Następnie, już w głównym modelu symulacyjnym, bootstrap zastąpiono źródłem napięciowym (podłączonym do V_{h+}), dającym napięcie o 6V wyższe niż napięcie zasilania:



Rysunek 25: Uproszczony układ symulacyjny ze sterowaniem górnej bramki analogicznym do BlueESC



Rysunek 26: Przebieg napięcia pomiędzy bramką a drenem tranzystora górnego dla układu jak na rys. 24



Rysunek 27: Układ sterujący górną bramką, ze źródłem napięciowym zamiast układu bootstrap

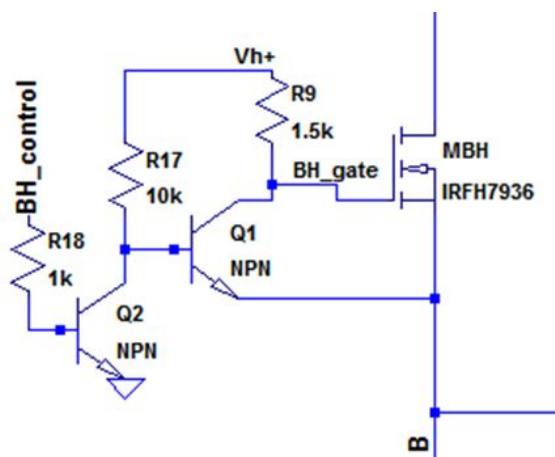


Rysunek 28: Przebieg napięcia pomiędzy bramką a drenem tranzystora górnego dla układu jak na rys. 27

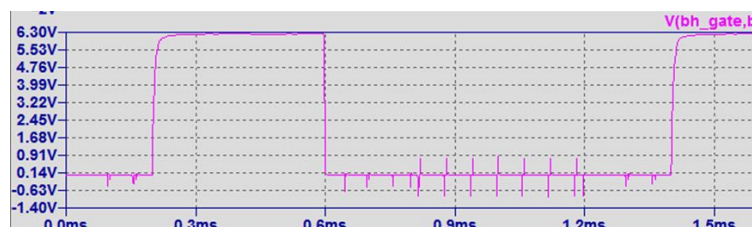
Jak widać, negatywne napięcie ponownie zbliżone jest to wartości napięcia zasilania (12V). Jednak dodatnie napięcie bramki jest bliskie 6V, czyli dokładnie wartości, jaką będzie dodawał do zasilania dodatkowy układ.

Kolejnym krokiem, mającym wyeliminować zależność negatywnego napięcia V_{GS} od napięcia zasilania, było odniesie tranzystora Q1 do napięcia fazy, zamiast do uziemienia.

Jak widać na rys. 30, ujemne napięcie na bramce prawie w ogóle nie występuje.



Rysunek 29: Układ sterujący górną bramką, ze źródłem napięciowym i odniesiem tranzystora Q1 do fazy



Rysunek 30: Przebieg napięcia pomiędzy bramką a drenem tranzystora górnego dla układu jak na rys. 29

Tym samym, zasilanie bramki jest już zupełnie niezależne od napięcia zasilania,

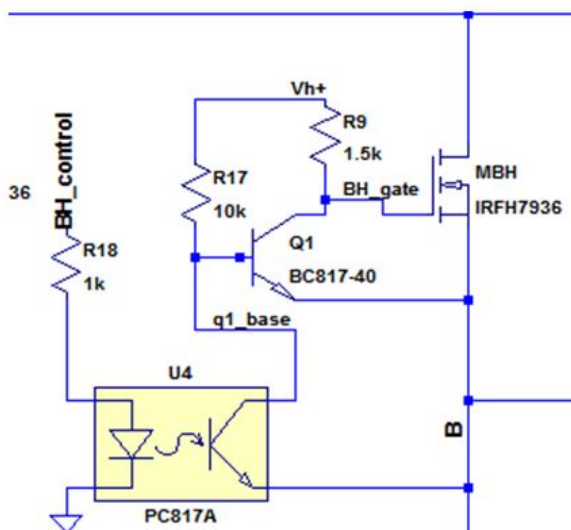
a więc układ w tej konfiguracji może pracować, teoretycznie, z dowolnie wysokim napięciem zasilania, a przy tym nie narażać bramki na przekroczenie dozwolonych napięć.

Układ w takiej formie został uznany za ostateczny i według takiego schematu wykonana została płytka PCB. Dopiero w czasie testowania prototypu okazało się, że w czasie projektowania został popełniony błąd, ponieważ układy sterowania górnymi tranzystorami nadmiernie się nagrzewały. Pomimo, że w symulacji zostało zweryfikowane, czy elementy nie są narażone na napięcia i wydzielane moce większe niż dopuszcza ich konstrukcja, przeoczono zweryfikowanie napięcia V_{EB} tranzystorów bipolarnych.

Otóż złącze EB tranzystora bipolarnego NPN, przewodząc z bazy do emitera zachowuje się jak zwykła dioda krzemowa, dając spadek napięcia ok. 0.7 V, a po zabezpieczeniu bazy odpowiednim rezystorem, kwestia napięcia V_{EB} nie musi już skupiać myśli projektanta. Nie można jednak zapomnieć, że przewodząc w odwrotną stronę, od emitera do bazy, złącze zachowuje się jak dioda zenera. Tym samym, gdy przekroczymy maksymalne napięcie V_{EB} , tranzystor zacznie przewodzić prąd w nieplanowany sposób, a towarzyszący temu spadek napięcia spowoduje szybkie przegrzanie tranzystora. W przypadku użytego elementu, przyłożone napięcie EB było znacznie wyższe niż maksymalne, podane w karcie katalogowej, wynoszące 6 V.

Żeby umożliwić jednak poprawne działanie gotowemu już sterownikowi, tranzystory bipolarne zmieniono na MOSFETy z kanałem typu N, umieszczone w tej samej obudowie SOT-23 i o analogicznych wyjściach. Po zmianie tranzystorów na płytce, układ działał poprawnie, ponieważ w takich zastosowaniach tranzystory bipolarne NPN i polowe z kanałem typu N zachowują się bardzo podobnie, z tą różnicą, że N-MOSFETy tolerują na analogicznych zaciskach znacznie wyższe napięcia (20, zamiast 6 V). Maksymalne napięcie pracy fizycznie zbudowanego sterownika, po modyfikacji nie może przekraczać 14V

Jeśli układ zasilania górnych bramek ma rzeczywiście działać niezależnie od napięcia zasilania całego sterownika, potrzebne jest także odizolowanie wejścia sygnału cyfrowego. Zaistniały błąd naprawiłaby zmiana tranzystora Q2 na transoptor, którego emiter byłby podłączony do potencjału fazy silnika.



Rysunek 31: Układ sterujący górną bramką, ze źródłem napięciowym i izolacją optyczną wejścia cyfrowego



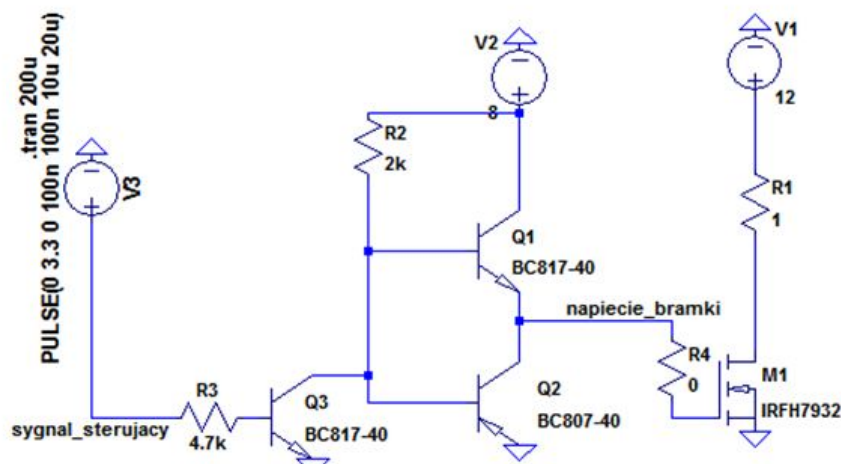
Rysunek 32: Przebieg napięcia pomiędzy bramką a drenem tranzystora górnego po zastosowaniu izolacji optycznej wejścia

Budując tego rodzaju układ, należy jednak pamiętać o opóźnieniu w przekazywaniu sygnału, występującym na transoptorach. Należy wybrać więc optoizolator o małym opóźnieniu.

3.4.2 Konstrukcja układów sterowania bramek tranzystorów dolnych

Zasilanie dolnych bramek jest znacznie prostsze, ze względu na to, że źródła tranzystorów mocy są połączone bezpośrednio z masą układu. Oznacza to, że wystarczy podać na bramkę tranzystora dolnego napięcie na przykład 5V, służące do zasilania mikrokontrolera czy innych układów połączonych ze sterownikiem. Zapewni to pełne otwarcie głównych tranzystorów, a do tego nie istnieje niebezpieczeństwa przekroczenia maksymalnego napięcia VGS. W niektórych sterownikach silników BLDC, sterowanie dolnymi bramkami jest maksymalnie uproszczone i wymaga tylko jednego lub dwóch rezystorów (na przykład w BlueESC). Do sterowania bramką wykorzystuje się bezpośrednio wyjście pięciowoltowego mikrokontrolera. Maksymalny ciągły

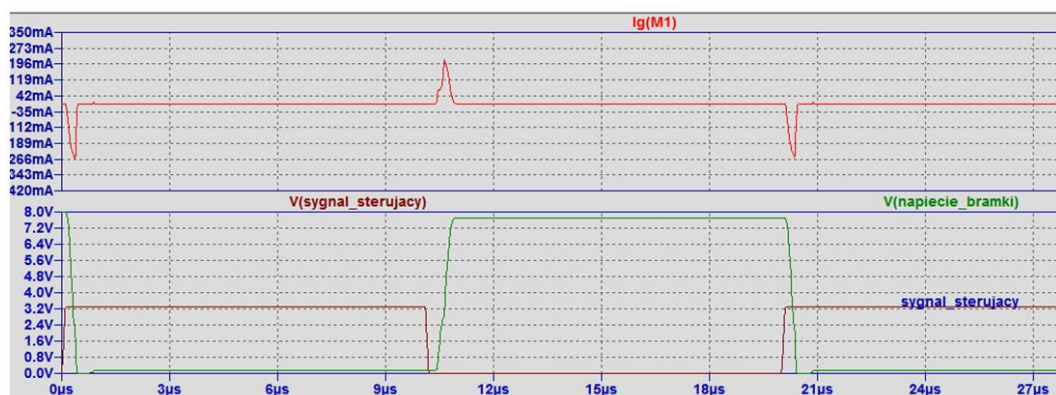
prąd wyjścia cyfrowego dla mikrokontrolera AVR ATmega168, często stosowanego w podobnych konstrukcjach, wynosi 20mA, w praktyce jednak chwilowo stosuje się większe prądy. To, w połączeniu z tranzystorami MOSFET o niskich ładunkach bramki i zmniejszoną częstotliwością kluczowania prądu, wystarcza do skutecznego sterowania tranzystorem. Dla symulacji identycznej jak w punkcie 1.1.3, ograniczenie prądu bramki do około 30mA, przy częstotliwości kluczowania prądu równej 16.7kHz, spowodowało wydzielanie 2.8W ciepła na tranzystorze mocy. To oznacza, że gdyby projektowany tutaj układ korzystał z innego mikrokontrolera, a maksymalny prąd na szynie prądu stałego wynosiłby skutecznie na przykład 25A, zasilanie dolnych bramek mogłoby być bardzo proste. Jednak ze względu na to, że zastosowany mikrokontroler pracuje na napięciu 3.3V, a także dlatego że wymagania co do obciążalności prądowej sterownika są dosyć wysokie, zostanie zaprojektowany osobny układ, zasilający bramki wyższym napięciem, ale przede wszystkim rozładowujący i rozładowujący bramki wyższym prądem. Oba cele realizują, bardzo popularne w elektronice, wzmacniacze mocy. Przykładem takiego układu może być wzmacniacz oparty o tranzystorach bipolarnych, w układzie przeciwsobnym (nazywany także push-pull, czyli pchać-ciągnąć). Wyjście tego układu może przewodzić znaczące prądy w obie strony, dlatego nadaje się do sterowania bramką tranzystora MOSFET, która elektrycznie jest kondensatorem, który w przebiegu sterowania silnikiem będzie naprzemiennie ładowany i rozładowywany.



Rysunek 33: Układ sterowania bramką dolną, z przeciwsobnymi tranzystorami bipolarnymi NPN

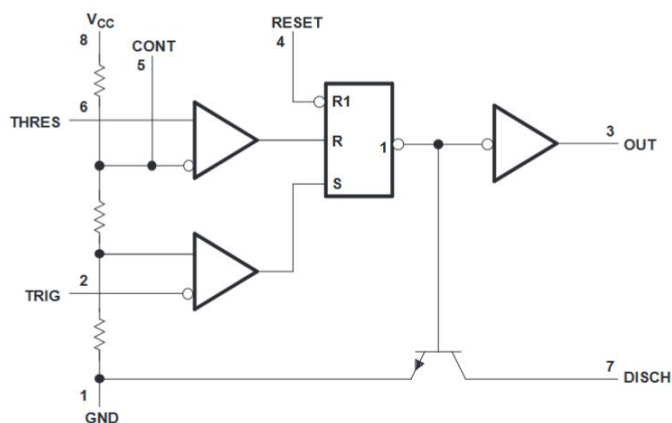
Jednak układ taki składa się z dużej ilości komponentów. Dodatkowo, projektowanie układu składającego się z wielu tranzystorów, bez posiadania bardziej rozległej wiedzy w tym temacie, wiąże się ze sporym ryzykiem popełnienia błędu. Dlatego postarano się skorzystać z gotowego układu scalonego, który ze względu na

swoją dostępność nie będzie łamać przyjętej zasady o nieużywaniu wyspecjalizowanych układów.



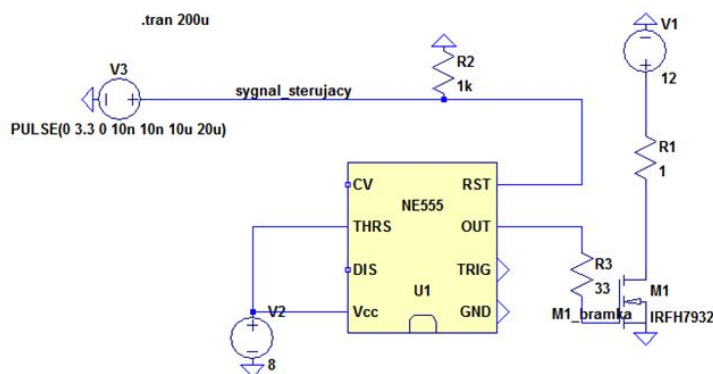
Rysunek 34: Przebiegi prądów bramki, napięcia sterującego i napięcia bramki, jak dla układu na rys.33

Zdecydowano się użyć układu NE555, jednego z najbardziej popularnych i wszechstronnych układów elektronicznych, jakie kiedykolwiek wyprodukowano. Jest on prosty w obsłudze, tani, bardzo dostępny, stabilny. Jest też produkowany przez wiele różnych firm, nie ma więc zbyt dużego prawdopodobieństwa szybkiego wycofania go ze sprzedaży. Jego wyjście jest wewnętrznie skonfigurowane jako układ push-pull, według karty katalogowej, zdolne przewodzić w sposób ciągły 200 mA, w obie strony. Układ ten można skonfigurować na wiele sposobów, by pełnił funkcję wzmacniacza sygnału cyfrowego. Wybrano jednak sposób najprostszy, nie wymagający użycia żadnych dodatkowych komponentów, poza rezystorem podciągającym wejście sygnału do masy i kondensatora na liniach zasilania.



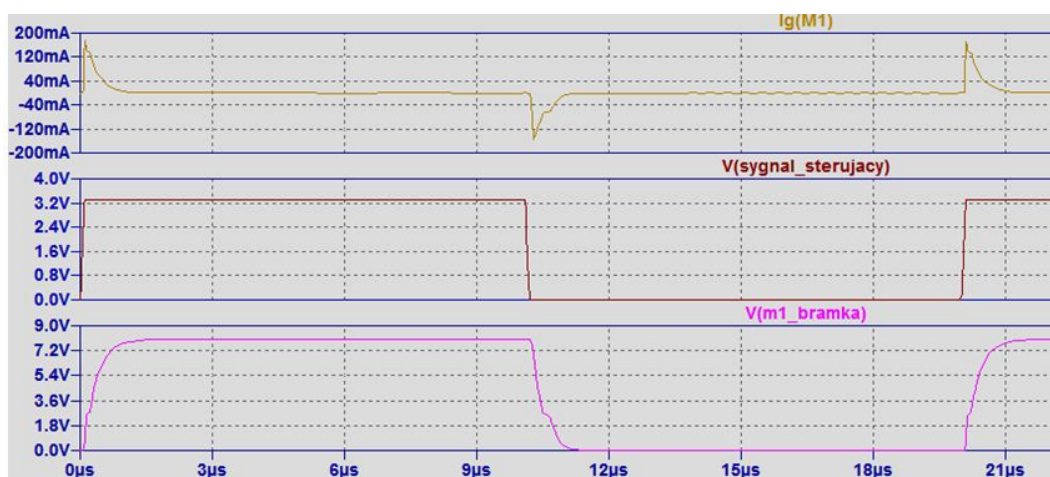
Rysunek 35: Uproszczony schemat działania układu NE555 z karty katalogowej

Wykorzystano tutaj wejście reset, którego podciągnięcie do masy zawsze skutkuje sygnałem 0 na wyjściu. Reszta układu została skonfigurowana tak, żeby wyjście miało wartość 1, zawsze kiedy na wejściu reset podane będzie dodatnie napięcie. Wewnętrznie, za wejściem reset, znajduje się bramka tranzystora bipolarnego, więc każde napięcie o wartości większej niż około 0.7 V będzie skutkowało stanem wysoki na wyjściu OUT.



Rysunek 36: Schemat sterowania bramką dolną, z wykorzystaniem układu NE555

Układ przetestowano w symulacji, zwracając uwagę w szczególności na prądy bramki tranzystora oraz napięcie bramki.



Rysunek 37: Zestawienie prądu bramki, sygnału sterującego i napięcia bramki tranzystora M1

Prąd bramki, ograniczony przez rezystor R3, wynosi maksymalnie około 160mA w obie strony. Dodatkowo, porównano czas jaki musi upłynąć od pojawienia się

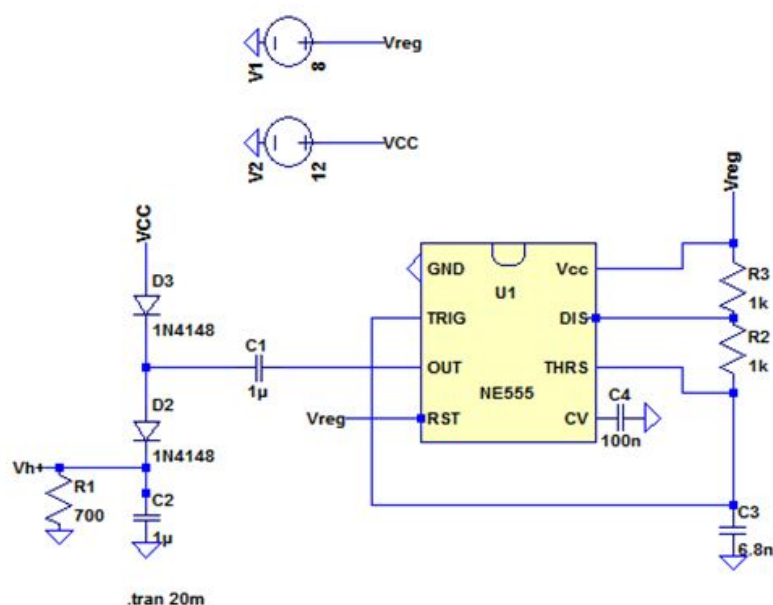
zbocza na sygnale sterującym do końca plateau Millera (czyli do momentu przeładowania pojemności bramki). Dla przedstawionego w tym rozdziale układu push-pull było to około 300-400 ns (różnie dla włączania i wyłączania), a dla układu sterującego z NE555 około 100 ns. Widać więc kolejną zaletę układu scalonego – jego komponenty są lepiej dobrane pod względem szybkości działania.

3.5 Układy pomocnicze

3.5.1 Generowanie podwyższonego napięcia

Układy zasilania bramek górnych tranzystorów wymagają do działania wygenerowania napięcia wyższego niż zasilanie układu. Jak wcześniej opisano, generowania napięcia wyższego o stałą wartość przez niezależny układ, posiada istotne zalety.

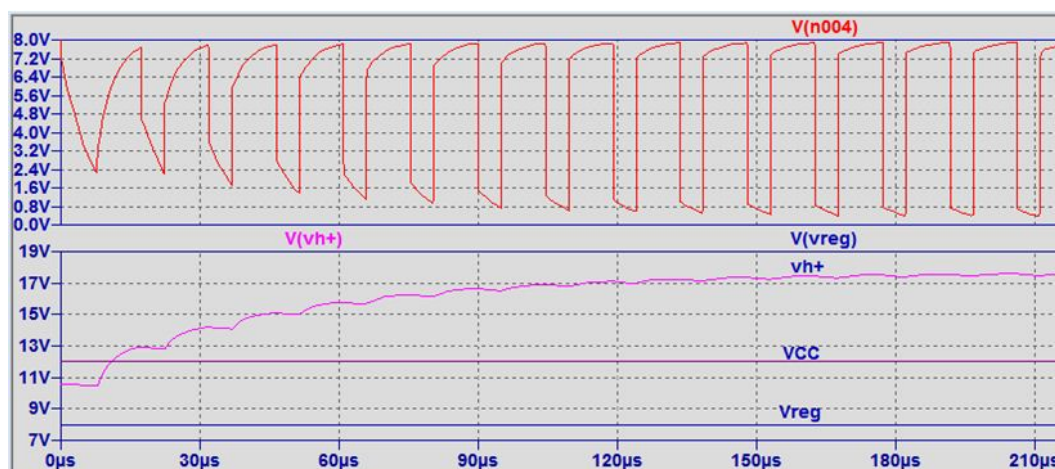
Układ zrealizowano korzystając z zasady działania kondensatorowych pomp ładunkowych. Układ NE555, ten sam którego użyto do sterowania bramek dolnych, generuje niezależnie na swoim wyjściu sygnał prostokątny. Jest on zorientowany w typowym układzie astabilnym, zgodnie z opisem zawartym w notce katalogowej.



Rysunek 38: Schemat układu generującego wyższe napięcie, składającego się z NE555 oraz pompy ładunkowej

Układ NE555 pracuje na niskim napięciu 8V, generowanym przez regulator liniowy (Vreg), znajdujący się na płytce. Jest odizolowany od wyższych napięć poprzez kondensator C1, znajdujący się na jego wyjściu. Generowany przez układ sygnał prostokątny ma częstotliwość około 80kHz. Ostateczny ładunek gromadzony

jest na kondensatorze C2, a napięcie na jego zaciskach (V_{h+}) jest wyjściem i źródłem podwyższonego napięcia. Na poniższych wykresach zaprezentowano napięcie w istotnych miejscach układu zaraz po jego uruchomieniu.



Rysunek 39: Przebiegi napięć wyjścia OUT NE555(n004), wyjścia pompy ładunkowej(V_{h+}), niższego napięcia, zasilającego NE555(V_{reg}) oraz zasilania całego sterownika (V_{CC})

Jak widać, podwyższone napięcie, pod obciążeniem, ustala się na poziomie poniżej 18V, co jest równe sumie V_{reg} i V_{CC} , pomniejszonej o spadki napięcia na elementach. Z racji że układ pracuje na napięciu regulatora umieszczonego na płytce, będzie podwyższał napięcie zawsze o stałą wartość, niezależnie od napięcia zasilania całego sterownika.

3.5.2 Generowania obniżonego napięcia

Układy zasilania dolnych bramek a także układ pompy ładunkowej wymagają niższego napięcia, najlepiej na poziomie około 7 V, tak by zapewnić założone wcześniej napięcie bramek tranzystorów mocy na poziomie 4.5 V. Obniżone napięcie nie jest jednak parametrem krytycznym, możliwe jest użycie napięcia nieco niższego lub wyższego (maksymalne napięcie pracy dla NE555 to 15 V), należy jednak pamiętać, że maksymalne dopuszczalne napięcie pracy jest określone w przybliżeniu zależnością $V_{CC} + V_{reg} < 20$ V. Obniżone napięcie generowane przez sterownik może służyć także do zasilania mikrokontrolera(po dodatkowym obniżeniu) lub innych układów współpracujących ze sterownikiem, jak na przykład odbiornik radiowy.

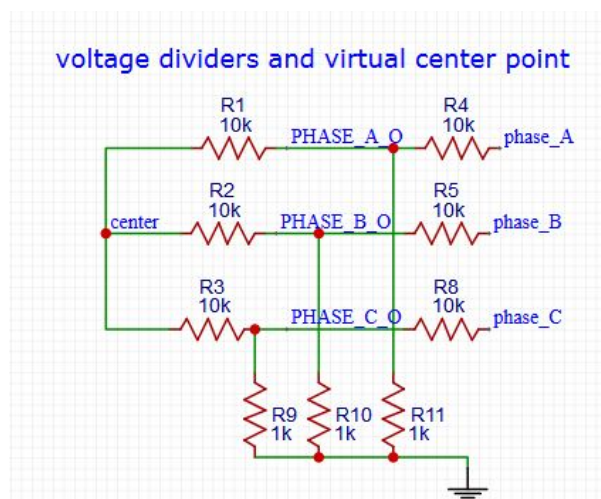
Najprostszym sposobem na uzyskanie niższego napięcia jest użycie regulatora liniowego. Są one bardzo proste w obsłudze, bardzo łatwo zaprojektować też układ z nimi związany. Wymagają najczęściej jednego lub więcej dodatkowych kondensatorów w celu ustabilizowania napięcia na ich wyjściu.

Oczywistą wadą regulatorów liniowych jest niska sprawność dla wyższych napięć.

Należy pamiętać o wydzielanej mocy, która jest równa iloczynowi spadku napięcia i prądu wyjściowego regulatora.

Jeśli byłoby potrzebne uzyskanie niskiego napięcia z potencjałów powyżej 20 V, istnieje możliwość wymontowania regulatora liniowego i podania niższego napięcia z zewnątrz, bezpośrednio dla układów NE555.

3.5.3 Dzielniki rezystorowe oraz sztuczny punkt neutralny



Rysunek 40: Schemat elektryczny dzielników napięcia i sztucznego punktu neutralnego. Wyjściami układu są węzły PHASE_x_O (gdzie x to oznaczenie fazy, A, B lub C) oraz center, wejściami punkty phase_x

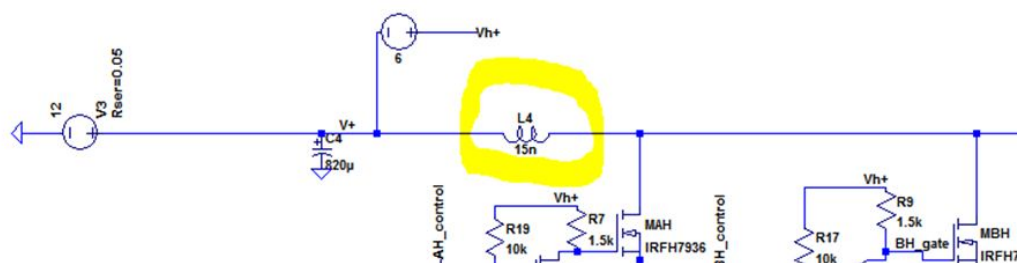
Ponieważ nie mamy dostępu do punktu środkowego, znajdującego się wewnątrz silnika, wykorzystano 3 rezystory (R1, R2, R3) w celu stworzenia sztucznego punktu środkowego. Wyjścia z faz A, B, C oraz punkt środkowy trafiają na wejścia przetwornika ADC mikrokontrolera, dlatego konieczne jest obniżenie ich napięcia. W tym celu, rezystory R4, R5 i R8 tworzą odpowiednio, razem z rezystorami R9, R10 i R11 dzielniki napięcia. Z dobranych wartości rezystorów wynika, że napięcie wyjściowe będzie stanowić około 9 procent napięcia wejściowego. Oznacza to, by nie przekroczyć napięcia 3.3 V dla wejścia mikrokontrolera, napięcie na fazie silnika może wynosić maksymalnie 36.6 V. Identyczna konfiguracja dzielników i punktu neutralnego została zastosowana w BlueESC.

3.6 Filtracja zasilania

Stosowanie kondensatorów w celu filtracji zasilania, szczególnie dla układów scalonych, jest powszechną praktyką. Ma ono na celu fizyczne przybliżenie źródła napięcia do zasilanego układu, co pozwala zmniejszyć impedancję pomiędzy źródłem a odbiornikiem. Kondensatory stanowią także zwarcie dla składowych o wysokiej

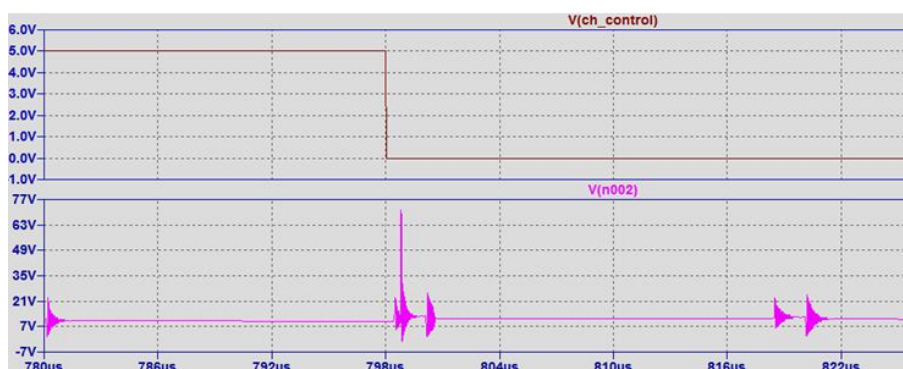
częstotliwości, które są niepożądane na zasilaniu. Z racji, że szczególnie w części mocy, występują bardzo wysokie natężenia prądu, a dodatkowo zmiany natężenia prądu następują bardzo szybko, układ narażony jest na skoki napięcia, powodowane przez pasożytnicze indukcyjności. Gdy prąd płynie przez indukcyjność i jego przepływ zostanie nagle przerwany, powstałe wtedy bardzo wysokie zmiany napięcia mogą niszczyć elementy. Naturę tego zjawiska dobrze oddaje angielskie określenie *inductive kick*, czyli indukcyjne kopnięcie.

By zasymulować zjawiska indukcyjne, pomiędzy drenami tranzystorów górnych a głównym kondensatorem filtrujący dodano indukcyjność o wartości 15 nH. Jest to wartość zbliżona do indukcyjności własnej przewodnika o wymiarach 0.5x5 mm oraz długości 30 mm, czyli do pocynowanej ścieżki na PCB, doprowadzającej prąd do górnych tranzystorów.



Rysunek 41: Umieszczenie cewki symulującej pasożytaniczą pojemność na schemacie układu

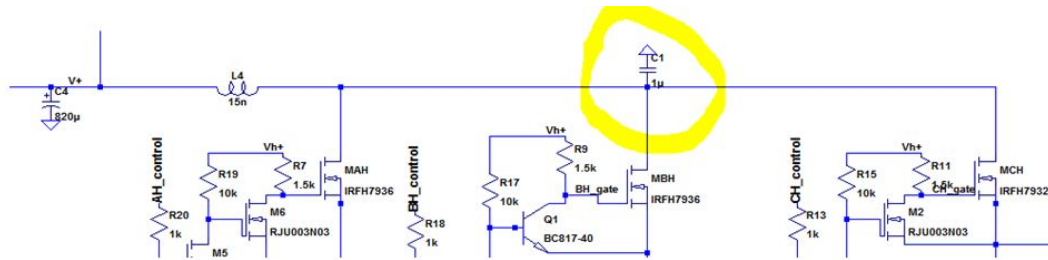
W wyniku symulacji, na potencjale drenów pojawiły się oscylacje o wysokiej częstotliwości i wysokiej amplitudzie. Na wykresie pokazano także napięcie sterujące bramką tranzystora CH, dzięki czemu widać, że największe oscylacje są skutkiem wyłączenia tranzystora CH. Jak widać, nawet indukcyjność niedługiej ścieżki PCB może skutkować zniszczeniem górnych tranzystorów.



Rysunek 42: Zestawienie sygnału sterującego tranzystorem CH i napięcia na źródłach tranzystorów górnych, z widocznymi indukcyjnymi skokami

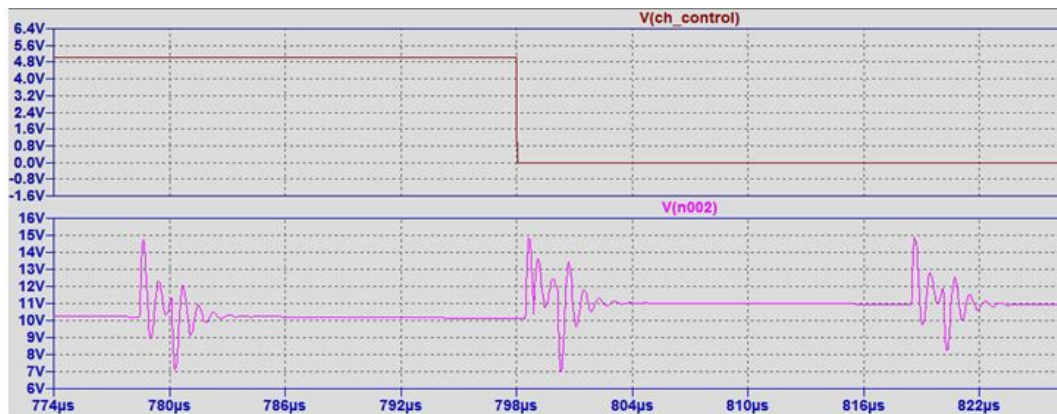
Następnie, w celu odfiltrowania nagłych skoków napięcia, bardzo blisko drenów

tranzystorów dodano kondensator ceramiczny o pojemności 1 μ F.



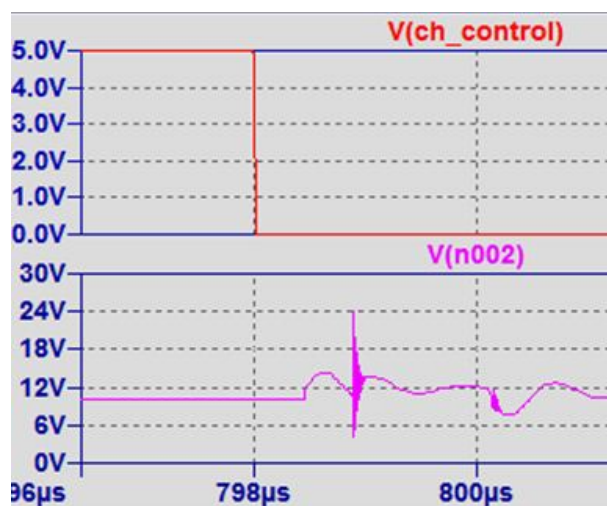
Rysunek 43: Umieszczenie kondensatora filtrującego na schemacie układu

Po przeprowadzeniu ponownej symulacji, widać znaczne zmniejszenie częstotliwości oraz amplitudy oscylacji.



Rysunek 44: Zestawienie sygnału sterującego tranzystorem CH i napięcia na źródłach tranzystorów górnych, po dodaniu kondensatora filtrującego

Pomimo bardzo obiecujących wyników, po wprowadzeniu do modelu kondensatora C1 indukcyjności pasożytniczej równej 800 pH (wartość typowa dla kondensatora 0.1 μ F, dla 1 μ F może być większa), zauważono istotny zwiększenie amplitudy oscylacji:



Rysunek 45: Zestawienie sygnału sterującego tranzystorem CH i napięcia na źródłach tranzystorów górnych, po dodaniu kondensatora filtrującego, z uwzględnieniem jego dodatkowej indukcyjności

W podobny sposób narażone na skoki napięcia narażone są także inne elementy układy, jednak rozważenie wszystkich przypadków nie jest możliwe w ramach tej pracy. Należy przyjąć zasadę, że ścieżki przewodzące prądy podlegające szybkim zmianom powinny być możliwie najkrótsze, a gdy to możliwe, powinny być filtrowane kondensatorami ceramicznymi, które mają mniejszą impedancję dla wyższych częstotliwości niż kondensatory elektrolityczne. Dodatkowo, maksymalne napięcie V_{DS} tranzystorów, jak również maksymalne napięcia innych elementów, powinny być dobierane z odpowiednim zapasem.

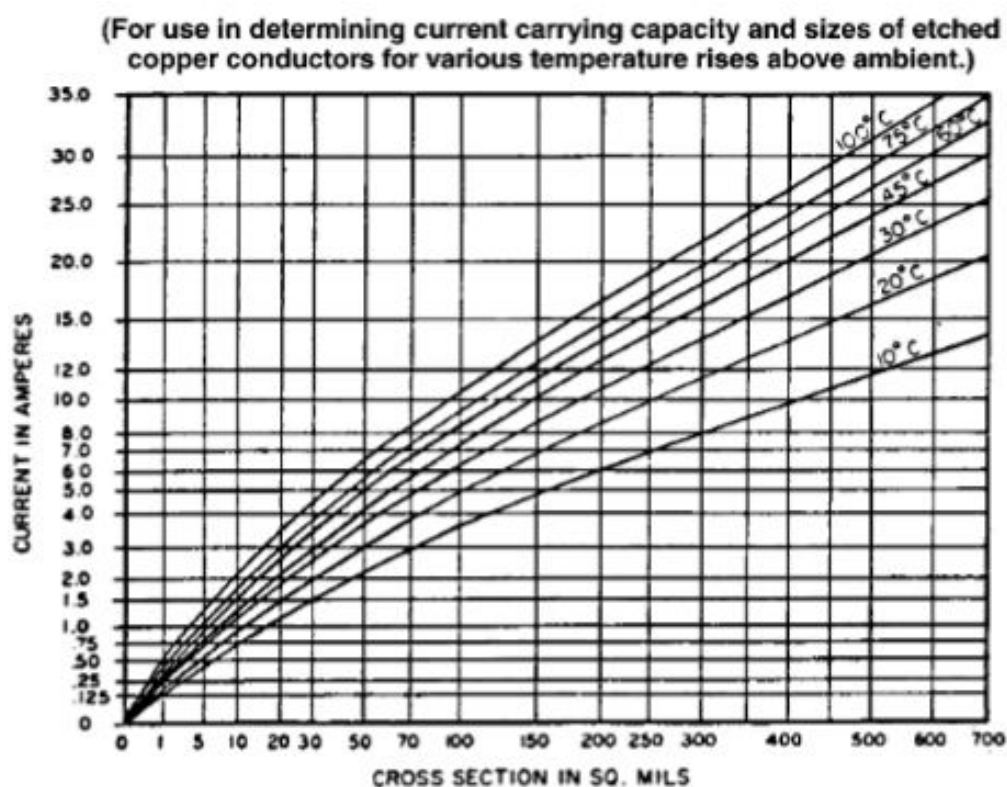
Na wejściu układu znajduje się także kondensator elektrolityczny, mający przede wszystkim gromadzić i oddawać energię, uśredniając tym samym prąd pobierany ze źródła zasilania i zmniejszając pulsację. Tak jak w poprzednim przypadku, pomaga on również uniknąć uszkodzeń, które bez kondensatora mogłaby wyrządzić stosunkowo duża indukcyjność przewodów zasilających. W symulacji, przez kondensator na wejściu płynie bardzo duży prąd, około 6 A RMS, przy użyciuiskoimpedancyjnego kondensatora elektrolitycznego. Zakupione kondensatory SamYoung NXH 470 μF 35 V LOW-ESR 10x16, mogą przewodzić w sposób ciągły 1.76 A. W podobnych konstrukcjach nie stosuje się jednak zazwyczaj więcej niż dwóch podobnych kondensatorów, można więc podejrzewać, że duży prąd wynika z właściwości symulacji, różniących ją od rzeczywistego układu (na przykład mniejsza indukcyjność uzwojeń silnika).

3.7 Płytki PCB

3.7.1 Wybrane zagadnienia dotyczące projektowania PCB

Wybór grubości ścieżek

Ze względu na wymagania producenta płytki, podstawową grubością ścieżki jest 8 mil. Przyjmując, że grubość warstwy miedzy wynosi około 1.4 mil (obecny standard w produkcji PCB), pole przekroju podstawowej grubości ścieżki wynosi około 11 mil^2 . Korzystając w wykresu (rys. 46), można odczytać że tej grubości ścieżka zewnętrzna może bezpiecznie i bez nadmiernego nagrzewania przewodzić prąd o natężeniu 1 A. Oznacza to, że dla prawie wszystkich ścieżek na płycie, biorąc pod uwagę głównie aspekt termiczny, zastosowania najmniejszej możliwej szerokości będzie wystarczające.



Rysunek 46: Krzywe ułatwiające dobór grubości miedzianej ścieżki PCB. Na osi pionowej - natężenie prądu w amperach, na poziomej - pole przekroju ścieżki w mil^2 , różne krzywe zostały narysowane dla różnych wzrostów temperatury ścieżki.

5

Wyjątkiem są tutaj ścieżki przewodzące prąd zasilający silnik. Ścieżki pomiędzy napięciem zasilania a źródłami tranzystorów górnych, jak i ścieżki prowadzące z dolnych tranzystorów do masy zasilania, będą przewodzić prąd o natężeniu około 30 A.

⁵[14] IPC-2221A, *Generic Standard on Printed Board Design*

Z wykresu można wywnioskować, że miedziany przewodnik dla takiego prądu powinien mieć mniej niż 700mil^2 powierzchni, a więc ponad 500mil (12.7mm) szerokości. Tak szeroka ścieżka, na dwuwarstwowej płytce, bardzo utrudniałaby rozplanowanie układu, oraz znacznie zwiększała jej rozmiar. W tym celu, ta ścieżka, oraz 3 inne, będące jednocześnie wyjściami faz, zostaną pozbawione maski lutowniczej oraz pocynowane, tak by wielokrotnie zwiększyć ich grubość. Ścieżka będącą masą całego układu, nie zostanie pocynowana w całości, musi być więc możliwie szeroka (preferowalnie szersza niż 1000mil).

Powyższe rozważania dotyczyła przewodzenia prądu stałego. Nie można zapomnieć, że większość ścieżek przewodzi prąd zmienny, przy którym trzeba wziąć pod uwagę również dodatkowe reaktancje. W związku z tym, wiele z połączeń, szczególnie doprowadzających zasilanie do układów scalonych, zostało dodatkowo poszerzonych, a celu obniżenia ich indukcyjności.

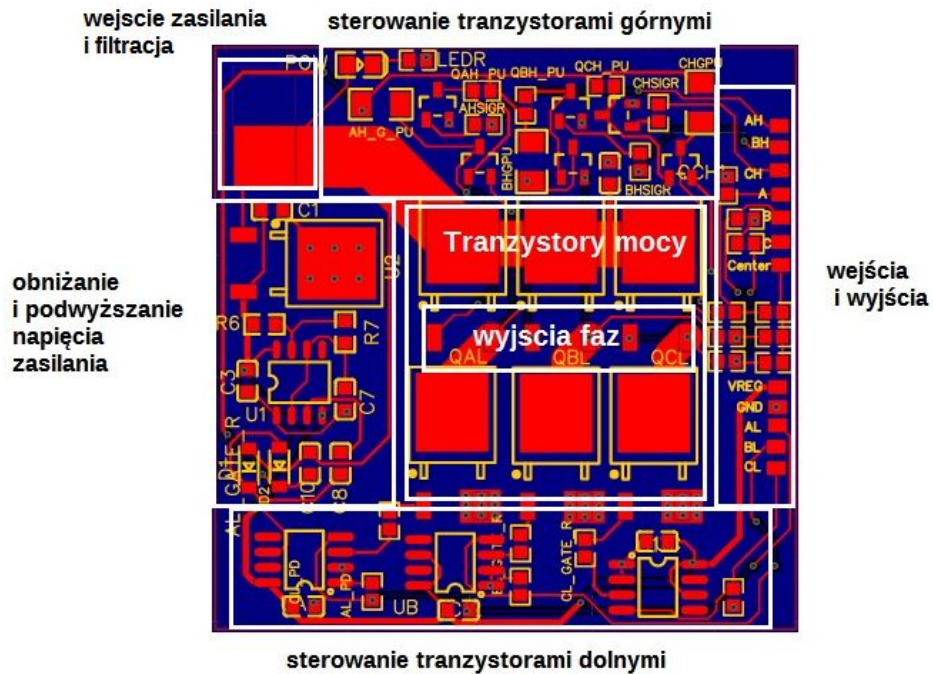
Otwory przewodzące

Projektując tor dla przepływającego prądu, nie można zapomnieć także o otworach na płytce, przewodzących prąd pomiędzy warstwami PCB (tzw. przelotki, ang. Via). Również one mogą posiadać znaczącą rezystancję i indukcyjność. Korzystając z kalkulatora ścieżek i przelotek firmy UltraCAD[16], wyliczono że jeden otwór o długości 3mm , średnicy 12mil oraz pokryty warstwą miedzy o grubości 1.4mil charakteryzuje się rezystancją $15\text{m}\Omega$, a jego oszacowana obciążalność prądowa wynosi 2.5A . Dane są te głównie przybliżone, ponieważ nie jest znana dokładny sposób wykonania otworów przez producenta płytki. W związku z tym, pomiędzy drenami dolnych tranzystorów a masą układu użyto 9 przelotek na każde połączenie, ponieważ natężenie prądu w tym miejscu wynosi około 20A RMS . Należy także pamiętać, że przelotki pomagają w odprowadzeniu ciepła z elementu na pole masy znajdujące się po drugiej stronie płytki. Głównie w tym celu, pod regulatorem napięcia umieszczono dodatkowe otwory.

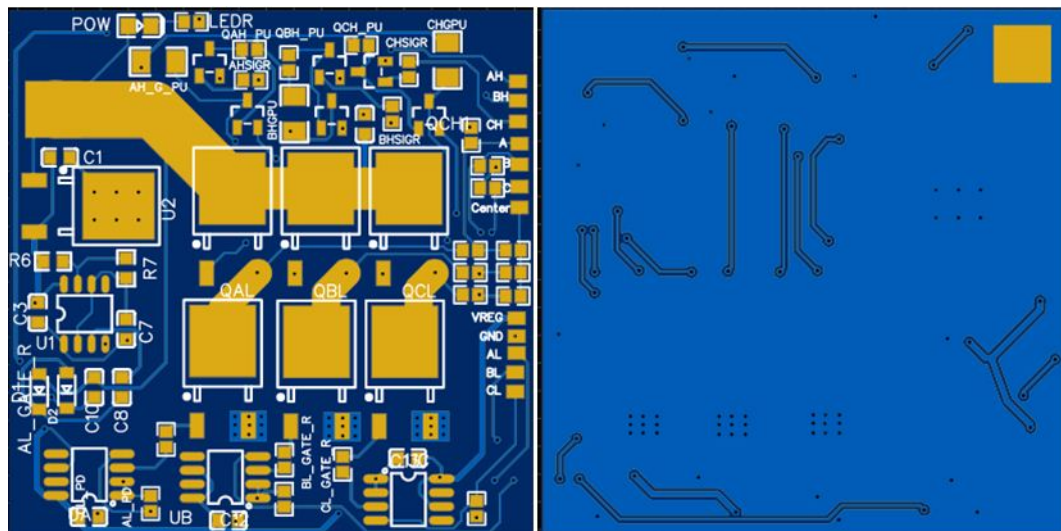
3.7.2 Przedstawienie wykonanej PCB

Płytkę zaprojektowano w środowisku EasyEDA, korzystając z przygotowanego w nim wcześniej schematu elektrycznego. Zazwyczaj tego typu konstrukcje opiera się na płytkach czterowarstwowych, umożliwiających umieszczenie części logicznej po jednej stronie, a części mocy po drugiej, jednocześnie oddzielając obie strony wewnętrznymi warstwami masy i zasilania. ‘Umożliwia to znaczne zmniejszenie fizycznych wymiarów sterownika, a także odizolowanie części mocy, będącej źródłem zakłóceń magnetycznych i elektrostatycznych. Projekt został jednak wykonany na płytce dwuwarstwowej, ponieważ są one znacznie tańsze i łatwiej dostępne niż czterowarstwowe. Ponadto, nie planowano umieścić mikrokontrolera na tej samej PCB, a nieco większy rozmiar nie stanowi problemu dla prototypu.

Układ na płytce został podzielony na 6 części, pełniących różne funkcje, zgodnie z ilustracją poniżej.



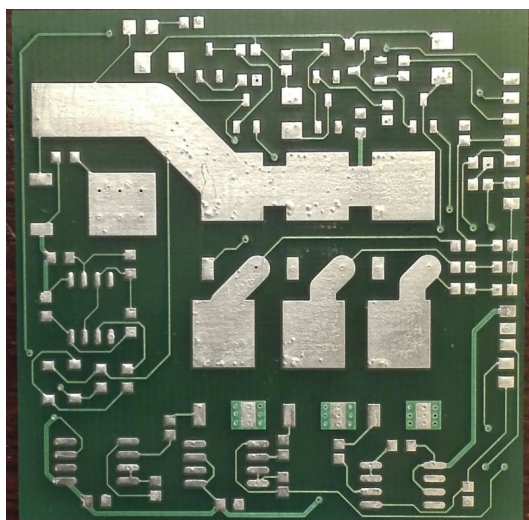
Rysunek 47: Schemat płytki PCB z podziałem na funkcjonalne części



Rysunek 48: Wizualizacja wykonanej płytki PCB. Z lewej znajduje się strona przednia, z prawej spódna

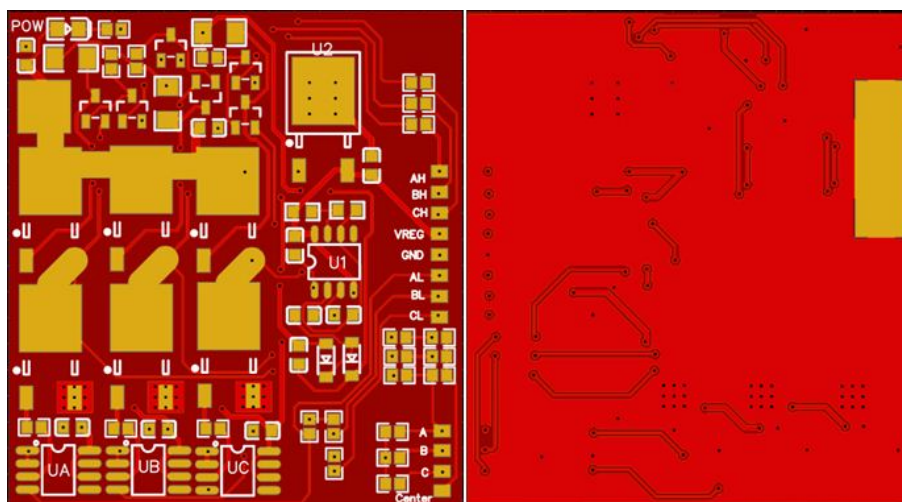
Poszczególne części funkcjonalne umieszczono tak, by zminimalizować długość i ułatwić tworzenie połączeń. Gotowa płytka ma wymiary kwadratu o boku około

48 mm. Według tego projektu została zamówiona i wykonana PCB do testowanego później prototypu. Spodnia strona została w całości wypełniona warstwą masy. Ma szereg zalet, między innymi ograniczenie przenoszenia się zakłóceń elektrostatycznych pomiędzy ścieżkami, skrócenie odległości jaką muszą pokonywać ładunki, oraz zapewnienie dużej szerokości ścieżki masowej.



Rysunek 49: Zdjęcie wykonanej płytki PCB

Słabymi punktami okazały się być stosunkowo duże odległości, które musi pokonać prąd o dużym natężeniu, mało uporządkowany rozkład wewnątrz niektórych bloków, oraz górna warstwa nadruku, nie nadająca się do wykonania (nie było to istotne, gdyż prototyp miał zostać wykonany bez niej). Dlatego zdecydowano się zaprojektować kolejną wersję. Nowa płytką ma wymiary 44x48 mm. Postarano się przede wszystkim, by maksymalnie zmniejszyć odległości pomiędzy tranzystorami a podłączeniem zasilania i kondensatorami. Bardziej uporządkowano ułożenie elementów, co pozwoliło zmniejszyć wymiar poziomy płytki o 4mm. Dodano miejsce na dodatkowy kondensator ceramiczny na wejściu układu i drugi kondensator elektrolityczny. Kolejna ma też przygotowany, poprawny nadruk, który może ułatwić korzystanie z płytki.



Rysunek 50: Wizualizacja drugiej wersji płytki

4 Program sterujący

4.1 Użyty mikrokontroler

W projekcie użyto mikrokontrolera STM32F407VGT6, umieszczonego na PCB stworzonym przez pracowników Politechniki. Mikrokontroler był używany razem z urządzeniem umożliwiającym programowanie i debuggowanie kodu, ST-Link V2.

Istotne dla projektu cechy mikrokontrolera:

- zasilany jest napięciem 3.3 V
- należy do serii F4, określanej przez producenta jako wysokowydajna
- procesor jest taktowany częstotliwością do 168 Mhz, wykonuje do 210 milionów operacji na sekundę
- posiada wyspecjalizowany koprocesor, służący do wykonywania operacji zmiennoprzecinkowych
- zawiera do 17 sprzętowych układów czasowych(timerów), 16 i 32-bitowych, pracujących z częstotliwością do 168 Mhz
- trzy, 12-bitowe przetworniki ADC są w stanie przetworzyć nawet 2.4 miliony próbek na sekundę (lub nawet 7.2 miliona próbek, korzystając ze wszystkich przetworników jednocześnie)
- zawiera 2, 12-bitowe przetworniki DAC
- posiada do 15 interfejsów komunikacyjnych, w tym trzy USART, sześć SPI, trzy I^2C , dwa CAN, SDIO oraz USB

Szczególnie ważne dla tego typu projektu jest częstotliwość taktowania procesora i liczników, odpowiednia funkcjonalność timerów (szczególnie PWM), oraz szybkość i rozdzielczość przetwornika ADC.

4.2 Użyte oprogramowanie

4.2.1 TRUEStudio

Jest zintegrowanym środowiskiem programistycznym, opartym na Eclipse, stworzonym przez firmę Attolic, zależną od STMicroelectronics. Współpracuje z wieloma mikrokontrolerami i płytkami ewaluacyjnymi STM. Posiada wiele przydatnych programistów narzędzi, jednak najistotniejszym jest wbudowany debugger o szerokich możliwościach. Poza podstawowymi funkcjami, takimi jak zatrzymywanie programu i podglądanie zmiennych, umożliwia ciągle przesyłanie zmiennych i wyświetlanie ich w formie wykresów. Dla wielu programistów taka funkcjonalność może wyeliminować potrzebę użycia oscyloskopu przy badaniu działania kodu.

4.2.2 STM32CubeMX

Jest programem od firmy STMicroelectronics, umożliwiającym konfigurowanie sprzętu mikrokontrolera w przyjaznej dla użytkownika formie graficznej. Umożliwia na przykład konfigurację częstotliwości zegarów i magistrali taktujących, ustawianie przeznaczenia wejść i wyjść cyfrowych, konfigurację timerów czy przetworników analogowo cyfrowych.

Program generuje kod, zawierający opis konfiguracji i funkcji inicjalizujący użyte peryferia sprzętowe. Generuje plik projektu, który może być używany w wielu różnych środowiskach programistycznych, między innymi w TRUEStudio. W gotowym projekcie kod podzielony jest na bloki generowane przez program i na kod użytkownika. Kod generowany przez STM32CubeMX używa bibliotek HAL (warstwa abstrakcji sprzętowej, ang. hardware abstraction layer).

4.3 Ogólna idea kodu

Główny kod, zarządzający pracą całego sterownika, znajduje wewnątrz funkcji main. To tam ustawiany jest aktualny tryb pracy, włączany jest rozruch i komutacja. Pozostałe zadania, głównie te, które muszą działać w czasie rzeczywistym, oparte są o przerwania. Dotyczy to głównie wykrywania przejść przez 0 w sygnale z faz silnika, włączania i wyłączania kroków komutacji, czy programu odpowiadającego za regulację prędkości obrotowej.

Ważne zadania, wywoływane z przerwai, wykonywane są głównie w czasie kroku 1 i 4, co sprawia, że zadania czasu rzeczywistego zawsze będą zajmować mniej niż $\frac{1}{3}$ czasu procesora.

4.4 Najważniejsze elementy sprzętowe służące do realizacji programu

W przypadku programowania mikrokontrolerów, należy możliwie dużo zadań przekazywać wyspecjalizowanym układom, znajdującym się wewnątrz urządzenia. Zazwyczaj ich zadaniem jest zmniejszenie obciążenia procesora i polepszenie terminowości wykonywania zadań. Rozbudowują one też funkcjonalność mikrokontrolera jeśli chodzi o sygnały analogowe.

4.4.1 Układy czasowe

W projekcie wykorzystano jedynie 3 układy czasowe, głównie ze względu na rozbudowane możliwości timera czwartego. Posiada on 4 kanały PWM, co oznacza, że wszystkie potrzebne wyjścia PWM można zrealizować korzystając z jednego timera. Do odliczania czasu kroku użyto timera 10, do cyklicznego wywoływania regulatora prędkości – timera 13. Oba pełnią tylko podstawowe funkcje, więc można w ich miejsce użyć dowolnego innego, dostępnego w mikrokontrolerze układu czasowego.

- Timer 4

Jak wcześniej wspomniano, timer ten generuje wszystkie 3 wyjścia PWM, używane do sterowania dolnymi tranzystorami. W tym celu, pierwsze 3 kanały zostały ustawione w tryb generowania PWM na wyjściu GPIO. Czwarty, nieużywany kanał został użyty do wywoływania konwersji ADC. Główny licznik, będący podstawą działania kanałów PWM, został skonfigurowany w następujący sposób: Prescaler (PSC) został ustawiony na 7 (wartość 16-bitowa) Okres licznika (ARR) – wstępnie na 399, jednak ponieważ częstotliwość sygnału PWM jest zmieniana w trakcie działania programu, rejestr ten jest modyfikowany. Pozostałe parametry, na jak przykład kierunek zliczania, zostały pozostawione jako wartości domyślne. Źródłem zegara dla timera 4 jest magistrala APB1. Według konfiguracji w STM32CubeMX, APB1 taktuje układ czasowy z częstotliwością 84 Mhz. Licznik jest więc inkrementowany co 12 Mhz (częstotliwość magistrali podzielona przez prescaler). Jeśli więc rejestr ARR jest ustawiony na 399, oznacza to że licznik będzie przepełniał się z częstotliwością 30 kHz (12 Mhz podzielone przez 399+1). Parametry te są dobrane tak, by dla wszystkich używanych częstotliwości zachować minimalnie ośmiobitową rozdzielczość sygnału PWM.

- Timer 10

Licznik tego układu czasowego jest taktowany z magistrali APB2, dla której częstotliwość zegara timerów wynosi 168 MHz. Dla uproszczenia programu uznano, że podstawą odmierzenia czasu przez timery będzie 1us. Dlatego, po ustawieniu prescalera na 168, licznik będzie inkrementowany z częstotliwością

1 MHz, a więc co 1us. Oznacza to, że czas w us, który będzie miał odmierzyć timer, będzie można wpisywać wprost do rejestru ARR, bez skalowania wartości. Licznik jest 16-bitowy, a więc będzie mógł odmierzyć maksymalnie 65536 us, czyli ponad 65ms, co zdecydowanie wystarcza, nawet dla najwolniejszej komutacji silnika.

- Timer 13

Został skonfigurowany w ten sam sposób co timer 10. Jest on jednak taktowany z APB1, co należy wziąć pod uwagę przy ustawianiu prescalera. W tym wypadku wynosi on 84. Ustalono, że regulator prędkości ma się wywoływać z częstotliwością 1 kHz, a więc rejestr ARR należy ustawić na 999.

4.4.2 Przetworniki analogowo-cyfrowe i cyfrowo-analogowe

Zamiana informacji analogowej na cyfrową jest niezbędna do działania algorytmu bezczujnikowej komutacji. Dlatego w projekcie wykorzystano przetwornik analogowo-cyfrowy ADC1, który dokonuje pomiarów z dwóch kanałów. Bardzo przydatny okazał się także przetwornik cyfrowo-analogowy.

- ADC

Maksymalną, możliwą do ustawienia, częstotliwością pracy ADC jest 21 MHz ($84 \text{ MHz} / 4$). 12-bitowy pomiar składa się z 12 cykli pomiaru raz 3 cykli próbkowania. – 1 pomiar trwa 0.54 us. Można więc założyć, że razem z wywołaniami, przesłaniem danych po DMA, pomiar dwóch kanałów trwa maksymalnie 1.5 us. Ustawienie 12-bitowej dokładności jest istotne, ponieważ dla niskich prędkości obrotowych, sygnał na wejściach mikrokontrolera może mieć jedynie kilkadziesiąt miliwoltów amplitudy. Przetwornik posiada kilka trybów działania i wywoływania. W projekcie został skonfigurowany w trybie scan conversion. Po wywołaniu konwersji ADC, w jednym ciągu wykonywane są konwersje z wielu zaprogramowanych kanałów (w tym przypadku z dwóch). Po wykonaniu konwersji wywoływane jest przerwanie, a wyniki pomiarów przesyłane są z wykorzystaniem DMA, wprost do pamięci, bez potrzeby angażowania procesora.

- DAC

Nie wymaga większej konfiguracji. Wywołania ustawiające poziom wyjścia przetwornika były wywoływane z poziomu kodu. Przetwornik nie pełni żadnej funkcji w gotowym programie, jego rola była jednak istotna podczas tworzenia go. O ile debugger wewnątrz TRUEStudio pozwala podejrzec zmienne w czasie rzeczywistym, jednak jego możliwości są ograniczone, zarówno co do szybkości jak i ilości przesyłanych danych. Przesyłanie zmiennych z częstotliwością kilkudziesięciu kiloherców obciążało środowisko po stronie komputera,

więzało się też z dużą ilością zgubionych/ nieprawidłowych danych. Przetwornik DAC pozwalał jednak w bardzo prosty sposób przesłać dane analogowe na wyjście mikrokontrolera, a następnie do oscyloskopu. Było to szczególnie pomocne przy testowaniu filtracji cyfrowej oraz przy podglądaniu przebiegów uzyskanych z ADC.

4.5 Najważniejsze funkcje realizujące zadanie komutacji silnika

Główny kod, zarządzający pracą całego sterownika, znajduje wewnątrz funkcji *main*. To tam ustawiany jest aktualny tryb pracy, włączany jest rozruch i komutacja. Pozostałe zadania, głównie te, które muszą działać w czasie rzeczywistym, oparte są o przerwania. Dotyczy to głównie wykrywania przejść przez 0 w sygnale z faz silnika, włączania i wyłączania kroków komutacji, czy programu odpowiadającego za regulację prędkości obrotowej.

4.5.1 Włączanie i wyłączanie kroków sekwencji

Za kontrolowanie tranzystorów wykonujących sekwencję odpowiada funkcja *step*, która włącza lub wyłącza odpowiednie tranzystory. Przekazana zmienna *step* zawiera informację o numerze kroku, który ma zostać zmieniony, a *state* o jego żądanym stanie. I tak wywołanie *step(3,1)* spowoduje włączenie odpowiednich tranzystorów przypisanych do kroku trzeciego. Funkcja ta izoluje algorytm na wyższym poziomie od komend specyficznych dla mikrokontrolera. W naszym przypadku wykorzystywane są funkcje biblioteki HAL, jednak równie dobrze na ich miejscu mogłyby się znajdować operacje dokonywane wprost na rejestrach mikrokontrolera, czy analogiczne komendy, odpowiednie dla innej platformy sprzętowej.

Listing 1: Skrócona prezentacja kodu funkcji *step*. Funkcja składa się z 6 case, które pominięto, ze względu na analogiczny kod

```
void step(int step, int state)                                1
{                                                            2
    switch (step) {                                          3
        case 1:                                             4
            if (state) {                                     5
                HAL_TIM_PWM_Start(&htim4, AL);              6
                //włączenie PWM na pinie AL                 7
                HAL_GPIO_WritePin(GPIOD, BH, HIGH);         8
                //Ustawienie stanu wysokiego na pinie BH   9
            }                                               10
        else{                                              11
            HAL_TIM_PWM_Stop(&htim4, AL);                   12
            HAL_GPIO_WritePin(GPIOD, BH, LOW);              13
        }                                                  14
        break;                                             15
        case 2:                                             16
```

```
        .
        .
        .
    }
}
```

17
18
19
20
21

4.5.2 Komutacja oparta na opóźnieniach programowych

Użycie tego typu algorytmu jest pomocne na etapie sprawdzania poprawności działania układu elektrycznego, ponieważ jest bardzo proste w implementacji. Finalnie funnkcja nie została użyta w oprogramowaniu sterownika, jednak przedstawienie jej jest pomocne w przedstawieniu sposobu włączania i wyłączania tranzystorów.

Listing 2: Skrócona prezentacja funkcji komutacja. Część instrukcji pominięto, ze względu na powtarzający się kod.

```
void komutacja(int Tc) {
    step(1,1); \\właczenie kroku pierwszego
    DWT_Delay(Tc); \\opoznienie przed wykonaniem kolejnej instrukcji o Tc
    step(1,0); \\wylaczenie kroku pierwszego

    step(2,1);
    DWT_Delay(Tc);
    step(2,0);

    .
    .
    .

    step(6,1);
    DWT_Delay(Tc);
    step(6,0);
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

Funkcja włącza więc krok, odczekuje odpowiedni czas (T_c , w mikrosekundach), następnie wyłącza go, następnie to samo czyni z kolejnym krokiem. Cała funkcja komutacja może być zredukowana do jednej pętli (nie uczyniono tego jednak dla większej czytelności). Po umieszczeniu wywołania komutacji w pętli `while`, mamy zapewnioną ciągłą pracę silnika, z zadany czasie jednego kroku.

Nie należy zapominać, że choć i tak prosty program pozwala skutecznie wprowadzić silnik w ruch, podczas jego działania nie należy się spodziewać uzyskania przebiegów napięć i prądów faz, typowych dla komutacji ze sprzężeniem zwrotnym.

4.5.3 Komutacja z wykorzystanie układu czasowego

Algorytm ten, ponieważ będzie aktywny równocześnie z innymi zadaniami mikrokontrolera, oraz będzie zależał od zdarzeń zewnętrznych (jak na przykład wykrycie przejścia przez 0 na sygnale z silnika), opiera się na przerwaniach, w celu odmierzenia czasu. Tak jak wcześniej, opóźnienie pomiędzy włączaniem a wyłączeniem

Listing 3: Skrócona prezentacja kodu funkcji odpowiedzialnej za komutację z użyciem timera, case od 2 do 5 pominięto, ze względu na analogiczny kod

```

if(htim->Instance == TIM10){           //Jeżeli przerwanie pochodzi od timera 10      1
    if(commutation_allowed || is_motor_starting_up)                                2
    {                                                                               3
        step(sequence_step, LOW); //wylacz aktualny krok                          4
        sequence_step++; //ziteruj krok                                           5
        if(sequence_step>6)                                                       6
            sequence_step=1;                                                       7
        conversion_per_step=0; /*zeruje zmienna przechowujaca informacje          8
            o ilosci konwersji wykonanych w czasie kroku*/                       9
        step(sequence_step, HIGH); //wlacz kolejny                               10
        HAL_TIM_Base_Stop_IT(&htim10); //zatrzymaj timer10                       11
        TIM10->CNT=0; //wyzeruj licznik timera                                    12
        TIM10->ARR =commutation_time; /*ustaw nowa wartosc                       13
            przepeelnienia timera, rwna okresowi trwania jednego kroku*/         14
        HAL_TIM_Base_Start_IT(&htim10); //w cz timer10                          15
    }                                                                               16
    else                                                                            17
        step(sequence_step, LOW);                                                 18
    }                                                                               19
}                                                                                   20
                                                                                     21
                                                                                     22
                                                                                     23

```

kroku było generowania poprzez komendę zajmującą procesor przez pewien czas, tak tutaj, za odmierzenie czasu będzie odpowiadał jeden ze sprzętowych timerów, który o upływie danego okresu poinformuje wywołaniem przerwania.

Czynności analogiczne do funkcji komutacja są wykonywane wewnątrz przerwania (HAL_TIM_PeriodElapsedCallback) wywoływanego po przepełnieniu się któregoś z liczników. Kod obsługujący przełączanie kroków nie został wydzielony do osobnej funkcji, w celu ograniczenia do minimum ilości instrukcji, które musi wykonać procesor wewnątrz przerwania. Użycie timera umożliwia ingerencję w moment kolejnej komutacji, pomiędzy kolejnymi przełączeniami. Jeśli układ sprzężenia zwrotnego stwierdzi, że następna komutacja powinna zostać na przykład opóźniona, może zatrzymać działający timer 10 i zmienić jego parametry.

Jeżeli przerwanie pochodzi od timera 10, wyłączany jest aktualnie trwający krok, włączony zostaje następny, a timer zostaje przygotowany, bo ponownie wywołał funkcję, gdy minie czas aktualnego kroku.

Po odpowiednim, wcześniejszym ustawieniu timera i ustawieniu flagi *commutation_allowed* (ang. komutacja dozwolona) lub *is_motor_starting_up* (czy trwa rozruch silnika) na wartość 1, algorytm ten będzie włączał kolejne kroki i zapewniał obracanie się silnika. Nie ma konieczności ani potrzeby ingerowania w komutację z poziomu głównej pętli programu. Zasilanie silnika będzie się więc odbywać niejako w tle, zostawiając czas procesora dla innych zadań.

4.5.4 Wykrycie przejścia przez zero

Cały kod algorytmu znajduje się wewnątrz funkcji *HAL_ADC_ConvCpltCallback*, która wywoływana jest po zakończonej konwersji ADC. Algorytm odejmuje od siebie uzyskane z konwersji napięcia, ponieważ znajdowanie przejścia przez 0 sygnału wymaga operowania różnicą napięć, pomiędzy punktem neutralnym a napięciem fazowym. Pierwszy warunek rozpoznaje przejście z wartości ujemnych na dodatnie. Zmienna *crossing_sign* zawiera informację o poprzednim znaku sygnału. Zależność od *sequence_step* sprawia, że warunek może zostać aktywowany tylko w czasie pierwszego kroku, ponieważ to w jego czasie powinno nastąpić szukane przejście. Zmienna *conversion_per_step* zawiera informację o tym, ile konwersji zostało wykonanych w czasie aktualnego kroku. Warunek z jej użyciem zapobiega wykryciu przejścia na początku kroku, gdy napięcie fazy jest jeszcze niestabilne. Po znalezieniu przejścia przez pierwszy if, *timer10* zostaje ustawiony tak, by wywołał zmianę kroku za pół czasu trwania kroku (*commutation_time*). Odjęcie stałej, równej 2 us powoduje kompensację opóźnień występujących w programie. Drugi warunek działa analogicznie, jednak wykrywa przejście z wartości dodatniej na ujemną. Dodatkowo, warunek ten służy do pomiaru okresu obrotu elektrycznego, z którego wyznaczany jest czas kroku. Jego obliczanie odbywa się w linii 18. i 20. *DWT->CYCCNT* jest strukturą biblioteki DWT, zawierającą informacje o ilości cykli, jakie wykonał procesor od włączenia mikrokontrolera. Wyznaczenie różnicy pomiędzy stanem cykli z aktualnego i poprzedniego wywołania i podzielenie jej przez 168 pozwala wyznaczyć różnicę czasu w mikrosendach.

Listing 4: Algorytm wykrycia przejścia przez 0

```

if (commutation_allowed && !is_motor_starting_up){           1
    voltage = voltages[1] - voltages[0];                      2
    if(crossing_sign==0 && voltage>1 && conversion_per_step>2   3
    && sequence_step==1)                                       4
    {                                                           5
        crossing_sign=1;                                       6
        HAL_TIM_Base_Stop_IT(&htim10);                        7
        TIM10->CNT=0;                                           8
        TIM10->ARR=(commutation_time/2) - 20;                 9
        HAL_TIM_Base_Start_IT(&htim10);                      10
    }                                                           11
    else if(crossing_sign==1 && voltage<-1 && conversion_per_step>2 12
    && sequence_step==4)                                       13
    {                                                           14
        crossing_sign=0;                                       15
        HAL_TIM_Base_Stop_IT(&htim10);                        16
        commutation_time_new=(DWT->CYCCNT-t1)/168;            17
        t1=DWT->CYCCNT;                                        18
        commutation_time= commutation_time_new/6;            19
        TIM10->CNT=0;                                           20
        TIM10->ARR=(commutation_time/2) - 2;                  21
        HAL_TIM_Base_Start_IT(&htim10);                      22
    }

```

```

    }
    conversion_per_step++;
}

```

23
24
25

4.6 Zmiana wypełnienia sygnału PWM

Wypełnienie sygnału PWM, sterującego dolnymi tranzystorami, jest zmieniane cały czas w trakcie pracy elektronicznego komutatora. By móc go wielokrotnie i bezpiecznie zmieniać, użyto następującego kodu:

Listing 5: Funkcja zmieniająca wypełnienie PWM

```

void PWM_changeDuty(float duty){
    if(!is_motor_starting_up)
    {
        global_duty=duty;
        compareVal=((timer4_period+1)*duty)-1; \\ timer4_period=360
        if(compareVal<170)
            chan4_compareVal=250;
        else chan4_comprarVal=110;
        new_duty=1;
    }
}

```

1
2
3
4
5
6
7
8
9
10
11

W celu zmiany wypełnienia, funkcja *PWM_changeDuty* jest wywoływana razem z wartością współczynnika wypełnienia, podanym jako liczba zmiennoprzecinkowa, o wartości od 0 do 1. Następnie wartość ta przepisywana jest do zmiennej globalnej. W kolejnej linii wyliczana jest potrzebna wartość rejestrów CCR(osignięcie jego wartości przez licznik wyłącza wyjście PWM) kanałów timera, tak by uzyskać zadane wypełnienie. Okres całego licznika ustawiony na 360. Dalej wybierana jest wartość CCR dla kanału czwartego, który steruje wywołaniem konwersji. Jeśli CCR dla kanałów odpowiedzialnych za PWM jest mniejsza od 170, dla kanału czwartego wynosi on 250, tak by konwersja została wykonana pewien czas po zakończeniu stanu wysokiego na dolnym tranzystorze. W przeciwnym wypadku, czyli jeśli CCR dla kanałów 1-3 jest większy od 170, CCR kanału czwartego zostaje ustawiony tak, żeby konwersja została wywoływana w około 1/3 okresu sygnału.

Listing 6: Zmiana wartości rejestrów odpowiedzialnych za wypełnienie PWM

```

if(new_duty){
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, compareVal);
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, compareVal);
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, compareVal);
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_4, chan4_compareVal);
    new_duty=0;
}

```

1
2
3
4
5
6
7
8

Sama operacja zmiany wartości rejestrów wykonywana jest w przerwaniu po wypełnieniu się licznika timera 4. Modyfikacja na rejestrów CCR na początku okresu

licznika, eliminuje szansę na wywołanie błędów w jego pracy. Rejestry timerów w STM32F407 nie są zabezpieczone przed zmienianiem ich w trakcie ich działania, a to oznacza że użytkownik swoją ingerencją może wywołać nieprzewidziane zachowanie się timera.

4.6.1 Wywołanie konwersji ADC

Przetwornik analogowo cyfrowy wywoływany jest poleceniem `HAL_ADC_Start_DMA(&hadc1, (uint32_t*)voltages, 2)` z poziomu programu. Fraza *DMA* odnosi się do tego, że po wykonaniu konwersji, wyniki zostaną przesłane wprost do przekazanych zmiennych (tablica *voltages*), za pomocą sterownika DMA.

Konwersja jest wywołwana z dwóch miejsc w kodzie – z przerwania po przepełnieniu się timera 4, oraz z przerwania *HAL_TIM_PWM_PulseFinishedCallback*, które wywołuje się po zakończeniu impulsu na kanale czwartym PWM. Wcześniej, w inicjalizacji, przerwania zależne od sygnału PWM zostały połączone z kanałem 4, ponieważ to on służy do pozycjonowania impulsu. Wywołanie po przepełnieniu timera 4 sprawia, że konwersja wykonywana jest na początku impulsu PWM. Doświadczalnie sprawdzono, że dokonywany w tym momencie pomiar jest użyteczny i pozbawiony większych zakłóceń. Zdarzało się jednak, że przy bardzo krótkich impulsach PWM (wypełnienie mniejsze niż kilkanaście procent), pomiary były zakłócone, co skutkowało pogorszeniem się jakości komutacji. Dlatego ustalono, że dla krótkich impulsów pomiar będzie dokonywany bliżej końca okresu sygnału PWM, z wykorzystaniem przerwania od zakończenia impulsu.

Przy wyższych wypełnieniach PWM, ADC wywoływane jest dwa razy na jeden okres sygnału. Było to konieczne, ponieważ przy wysokich prędkościach obrotowych, sprawdzanie wystąpienia przejścia przez zero tylko raz na okres okazało się być zbyt rzadkie do poprawnej komutacji. Ponieważ wywołanie konwersji po zakończeniu impulsu została zrealizowane z użyciem kanału PWM, szczegóły jego konfiguracji zostały opisane w podrozdziale *Zmiana wypełnienia sygnału PWM*.

Listing 7: Wywołanie konwersji ADC wewnątrz przerwania po przepełnieniu się timera 4

```
else if(htim->Instance == TIM4 ){                                1
    if(((sequence_step==1 && crossing_sign==0) ||                2
        (sequence_step==4 && crossing_sign==1)) && compareVal>70) 3
    {                                                            4
        HAL_ADC_Start_DMA(&hadc1, (uint32_t*)voltages, 2);      5
        conversion_per_step++;                                    6
    }                                                            7
}                                                                8
```

4.6.2 Rozruch

Algorytm rozruchu korzysta z komutacji z wykorzystaniem timera. Pracuje ona w tle i rozpoczyna się w momencie ustawienia flagi *is_motor_starting_up*. Następnie w pętli zmieniana jest wartość *commutation_time*, z której korzysta funkcja wykonująca komutację. Pomiedzy zmianami wartości wykonanie instrukcji jest opóźniane o czas, jaki powinien trwać aktualny krok. Gdy *j* osiągnie wartość 120, rozruch kończy się.

Współpraca pomiędzy funkcją start a komutacją nie jest ścisła w sensie czasowym, to znaczy zmiana czasu kroku (*commutation_time*) nie wywołuje natychmiastowego skutku w algorytmie komutacji, jednak w tym przypadku nie stanowi to problemu. Dla przykładu, jeśli funkcja rozruchu zostałaby z jakiegoś powodu opóźniona (np. poprzez pojawiające się przerwania), brak jej terminowości nie powinien znacząco wpłynąć na rozpędzanie się silnika.

Listing 8: Funkcja odpowiadająca za rozruch silnika

```
void modelarski_start()
{
    PWM_changeDuty(0.4); // wyłączenie PWM ustawiane jest na 40%.
    j = 0;
    sequence_step=1;
    is_motor_starting_up=1;
    while (j<120) {
        commutation_time= startTable[j];
        DWT_Delay(startTable[j]);
        j++;
    }
    is_motor_starting_up=0;
}
```

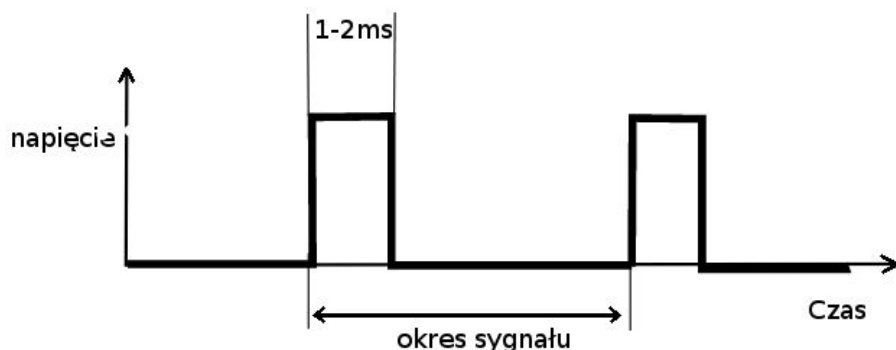
4.7 Pozostała funkcjonalność sterownika

4.7.1 Odbiór sygnału PWM

Sygnał PWM jest podstawowym sposobem komunikacji pomiędzy centralnym mikrokontrolerem a innymi urządzeniami na pokładzie modeli latających. Jeśli robot odbiera informacje od człowieka – prawdopodobnie dzieje się to za pośrednictwem aparatury modelarskiej, której odbiornik wysyła informacje zakodowane w sygnale PWM. Również serwomechanizmy i regulatory obrotów odbierają sygnał tej formie. Z tego powodu również projektowany regulator będzie odczytywał informacje zawarte w sygnale PWM.

Standardem w modelach latających jest kodowanie wartości poprzez nadanie impulsu o odpowiedniej długości. Typową częstotliwością sygnału jest 50 Hz, a długość impulsu leży zazwyczaj w zakresie 1-2ms. Najczęściej 1 ms oznacza wartość

najniższą, 2 ms – najwyższą.



Rysunek 51: Rysunek przebiegu typowego dla modeli latających sygnału PWM

W naszym regulatorze, w celu odczytania informacji z sygnału PWM, trafia on na jeden z pinów mikrokontrolera. Wejście cyfrowe jest skonfigurowane tak, że każde zbocze na nim wywołuje przerwanie. Następnie, wystarczy zbadać ilość czasu, upływającą pomiędzy zboczem narastającym a opadającym sygnału. Oznacza to, że dla programu nie ma znaczenia odległość pomiędzy kolejnymi impulsami – czyli nie ma znaczenia jego okres. W komercyjnych regulatorach prędkości odbywa się to w podobny sposób, co pozwala na dostosowanie częstotliwości do potrzeb platformy. W zastosowaniach, które nie wymagają częstych zmian mocy silników, używa się sygnału o częstotliwości 50 Hz, natomiast w wielowirnikowcach, których kontrolery lotu wywołują algorytmy stabilizujące parametry lotu kilkaset razy na sekundę, stosuje się sygnał o częstotliwości nawet 400 Hz.

Sygnał z odbiornika ma napięcie 5V, jednak większość pinów STM32F407 toleruje ten poziom napięcia jako wejście.

Listing 9: Kod przerwania zewnętrznego, mierzący długość impulsu odbieranego sygnału

```

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    1
    2
    3
    if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_1))//jesli wystapilo zbocze narastajace
    4
        t2=DWT->CYCCNT;
    5
    6
    7
    else { //jesli wystapilo zbocze opadajace
    8
        //wyznaczanie roznicy czasu
    9
        new_pulse_in_duration=(DWT->CYCCNT-t2)/168;
    10
    11
        if((new_pulse_in_duration<900)|| (new_pulse_in_duration>2100))
    12
            PWM_OK=0;
    13
    14

```

```

else
{
    PWM_OK=1;
    pulse_in_duration= new_pulse_in_duration;
}
if(PWM_OK&&commutation_allowed&&is_PWM_an_input)
    PWM_changeDuty(((pulse_in_duration-1000)/1000.0));
}
}

```

4.7.2 Algorytm regulacji prędkości obrotowej

Stworzony do tej pory sterownik dba, by generowane przebiegi odpowiadały aktualnej pozycji i prędkości silnika. To zjawiska fizyczne zachodzące w silniku decydują z jaką prędkością się on obraca.

Jeśli więc do pracującego silnika dołożymy obciążenie, wymagany będzie wyższy moment obrotowy. A ponieważ napięcie przyłożone do silnika się nie zmieni, wzrost momentu obrotowego dokona się kosztem prędkości. Jeśli więc chcemy mieć kontrolę nad prędkością obrotową, niezależnie od czynników zewnętrznych, musi zostać zaimplementowany algorytm, który będzie mógł zadać i utrzymać prędkość, modyfikując napięcie przyłożone do silnika.

W tym celu zostanie stworzony algorytm bazujący na zasadzie działania regulatora PI. Ustalono, że algorytm będzie się wywoływał z częstotliwością 1000 Hz. Do cyklicznego wywołania użyto przerwania od przepełnienia timera 13.

Kod funkcji znajduje się poniżej. Jego podstawowe funkcje to obliczenie różnicy pomiędzy przefiltrowaną prędkością silnika a prędkością zadaną, następnie sumowanie błędów, a potem ustalenie odpowiedzi regulatora i wpisanie jej do zmiennej *global_duty*. Pierwszy składnik sumy to część proporcjonalna, drugi to część całkująca. Oprócz tego wykrywane jest wejście w ograniczenia przez wyjście regulatora by ograniczyć nadmierne nagromadzenie błędów przez człon całkujący (zjawisko wind-up).

Listing 10: Kod algorytmu regulacji obrotów

```

else if(htim->Instance == TIM13){
    if (commutation_allowed&&!is_PWM_an_input&&!is_motor_starting_up) {
        commutation_speed= 1000000/(commutation_time*6);
        speed_filtered= filter_func(commutation_speed, 1);
        error = speed_filtered- set_commutation_speed;

        if(!is_controller_saturated)
            error_integral+=error; //całkowanie błędów

        is_controller_saturated=0;

        if(error_integral<-200000)
            error_integral= 200000;
    }
}

```

```

global_duty= -((error/100.0)+(error_integral/40000.0)
14
15
16
if(global_duty>0.9)//ograniczenia wyjscia
17
18
{
19
20
    global_duty=0.9;
    is_controller_saturated=1;
21
22
}
23
else if( global_duty<0.2)
24
25
{
26
27
    global_duty=0.2;
    is_controller_saturated=1;
28
29
}
30
PWM_changeDuty(global_duty);
}

```

Dobłą praktyką jest, by w przerwaniach umieszczać tylko niezbędne obliczenia, tak by wykonanie przerwania zajmowało możliwie mało czasu. Pomaga to w utrzymaniu terminowości i niezawodności programu. Z racji że kod regulacji prędkości wywołuje się stosunkowo rzadko a jego ścisła terminowość nie jest niezbędnie konieczna, część obliczeń powinna zostać przeniesiona do głównej pętli w funkcji *main*. Wykonanie kodu powinna się uzależnić od wystąpienia flagi, którą ustawiałoby cyklicznie przerwanie. Tutaj tego nie wykonano, ze względu na konstrukcję kodu testowego (zawiera ona opóźnienia programowe). Podobnie zostało to zrealizowane w UM1594, tam jednak cały kod regulatora prędkości znajdował się w pętli głównej.

4.7.3 Filtr cyfrowy

W regulatorze obrotów użyta funkcja *filter_func*. Przyjmuje ona jako argument aktualną wartość sygnału, następnie wykonuje obliczenia z jej użyciem, a następnie zwraca obliczoną wartość, która jest wyjściem filtra cyfrowego. Wewnątrz zastosowano filtr alfa-beta, który jest uproszczoną formą obserwatora, spokrewnioną z filtrami Kalmana. Główną zaletą użytego filtra jest łatwość implementacji. Jego użycie nie wymaga też szczegółowego modelu systemu. Pierwotnie miał on służyć do wygładzania napięcia fazy przy wykrywaniu przejścia przez zero, nie znalazł tam jednak zastosowania. W czasie testowania regulatora prędkości pojawiła się jednak potrzeba filtrowania jej, więc to tam użyto prezentowanego algorytmu.⁶

Listing 11: Kod filtra alfa-beta ⁷

```

int16_t filter_func(int16_t val, uint16_t channel)
1
{
2
    if(channel==1){
3
        ypri = ypost+dt*vpost;
4
        vpri=vpost;
5
    }
}

```

⁶filtru alfa-beta użyto w ten sam sposób w [8]Engine Speed Monitoring: The Alpha-Beta Filter

```

        ypost= ypri +alfa*((float)val-ypri);
        vpost = vpri + beta*((float)val - ypri)/dt;
        return ypost;
    }

```

4.7.4 Komunikacja z użyciem magistrali CAN

Sterownik ma mieć możliwość przesłania danych o aktualnej prędkości obrotowej silnika. Zostanie do tego użyta magistrala CAN, która pierwotnie została stworzona do zastosowania w pojazdach, lecz obecnie używana jest także w przemyśle. Składa się przeważnie z dwóch przewodów, oznaczonych jako CANH i CANL. Przesyłana informacja jest kodowana sygnałem różnicowym, co zapewnia dużą odporność na zakłócenia. Użyty mikrokontroler posiada specjalny układ odpowiedzialny za obsługę CAN. Pozostałe istotne cechy magistrali to:

- nie istnieje w niej wyobredniona jednostka centralna
- nadawane komunikaty odbierane są przez wszystkie podłączone urządzenia
- każdy wysyłany zestaw danych posiada identyfikator
- obsługa dostępu do magistrali odbywa się automatycznie. Wszystkie urządzenia przed wysłaniem komunikatu sprawdzają czy magistrala jest wolna.

Program zostanie zbudowany w oparciu o przykład dostępny w bibliotekach programu STM32CubeMX⁸.

Funkcja inicjalizująca CAN zostanie wygenerowana przez STMCube, zgodnie z wybranymi w konfiguratorze parametrami. Część niezbędnych czynności pozostanie jednak w gestii użytkownika. Należy przede wszystkim zadeklarować zdefiniowaną wewnątrz bibliotek HAL strukturę nagłówka komunikatu CAN i tablicę przechowującą przesyłane dane. Najbardziej istotne parametry nagłówka to:

- StdId – identyfikator standardowy, używany w protokole wersji 2.0A, składający się z 11-bitów.
- ExtId – przedłużony identyfikator, używany w wersji 2.0B
- RTR – specyfikuje typ ramki danych
- IDE – specyfikuje typ identyfikatora

⁷ Jest to zaadaptowana do języka C wersja kodu z [10] *Filtr Alfa – Beta od teorii do praktyki – 1*

⁸ przykłady znajdują się pod ścieżką STM32Cube\Repository\STM32Cube_FW_F4_V1.24.0\Projects\STM324xG_EVAL\Examples\CAN

- DLC –określa długość ramki danych, od 0 do 8 bajtów. W naszym przypadku przesłane zostaną dwa bajty. – możliwe jest zawarcie w ostatnich dwóch bajtach ramki danych sygnatury czasowej wysłania komunikatu. Parametr włącza lub wyłącza tę opcję

Listing 12: Deklaracje i przypisanie wartości parametrom nagłówka wiadomości

```
CAN_TxHeaderTypeDef TxHeader;      1
uint8_t TxData[8];                2
TxHeader.StdId = 0x321;            3
TxHeader.ExtId = 0x01;            4
TxHeader.RTR = CAN_RTR_DATA;      5
TxHeader.IDE = CAN_ID_STD;         6
TxHeader.DLC = 2;                  7
TxHeader.TransmitGlobalTime = DISABLE; 8
```

STMCube nie generuje automatycznie funkcji konfiguracji CAN(jedynie inicjalizację), można więc ją zaczerpnąć z dostarczonego przykładu. Po skonfigurowaniu transmisji, urządzenie obsługujące komunikację należy uruchomić funkcją `HAL_CAN_Start`.

Do nadania wiadomości należy użyć funkcji która zgodnie z dokumentacją funkcji, dodaje wiadomość do pierwszej wolnej skrzynki nadawczej(nazywa się tak miejsce, w którym dane oczekują na transmisję) i aktywuje żądanie nadania.

Zmienna przechowująca nadawane dane jest w przykładowym kodzie zadeklarowana w następujący sposób: `uint8_t TxData[8]`; Jak widac, talica składa się z ośmiu, ośmiobitowych elementów(bajtów). Przy przesłać zmienną 16-bitową (bo takimi w większości posługuje się kod sterownika), można ją podzielić na dwa bajty. Bajt starszy zostaje stworzony z użyciem przesunięcia bitowego, młodszy poprzez wyzerowanie pierwszych 8 bitów oryginalnej zmiennej, poprzez operację AND(&).

Listing 13: Sposób dzielenia zmiennej 16-bitowej na dwie, ośmiobitowe

```
TxData[0] = value >> 8;          1
TxData[1] = value & 0x00FF;      2
```

Bardzo podobna jest konfiguracja CAN do odbioru, z tą różnicą, że należy użyć przerwania `HAL_CAN_RxFifo0MsgPendingCallback`, które zostanie wywołane gdy odebrana zostanie transmisja.

5 Testy i wnioski

5.1 Układ elektryczny

Po stworzeniu układu, jako pierwsze została zbadana poprawność działania układów obniżających i podwyższających napięcie. Na wyjściu pompy ładunkowej, przy zasilaniu napięciem 12V, zmierzono około 16.8V. Napięcie to jest o ok. 1 V mniejsze niż zakładano ponieważ gdy sterownik nie pracuje, układ podwyższający napięcie jest maksymalnie obciążony, ponieważ 3 górne bramki są w tym momencie uziemiane. Układ obniżający napięcie działał bez zarzutów.

Następnie sprawdzono na oscyloskopie przebiegi generowane na bramkach wszystkich 6 tranzystorów mocy. Wszystkie były poprawne i zgodne założeniami. Niestety, po pierwszym złożeniu układu stwierdzono nadmierne nagrzewanie się zasilania górnych bramek. Usterkę tą poprawiono, co zostało już opisane przy omawianiu bramek górnych tranzystorów. Okazało się, że jeden z tranzystorów jest narażony na przekroczenie maksymalnego napięcia V_{GS} , co ogranicza maksymalne napięcie zasilania do około 13V. By układ rzeczywiście mógł pracować z napięciem wyższym niż 18V, potrzebna jest izolacja galwaniczna wejścia cyfrowego.

tranzystory mocy działały prawidłowo, w czasie testów nie nagrzewały się nadmiernie. Na wejściu układu i na kablach zasilających fazy dodano rezystory o wartości 0.1 Ohma, umożliwiające pomiar prądu. Wizualizacja natężenia prądu na wejściu układu pomaga stwierdzić, czy kroki włączają się z odpowiednim odstępem. Nawet bardzo krótkie zwarcie byłoby widoczne na przebiegu prądu regulatora. Układ pracował w czasie testów około 10h, przeważnie bez większego obciążenia. W tym czasie nie wystąpiła żadna awaria.

Wnioski:

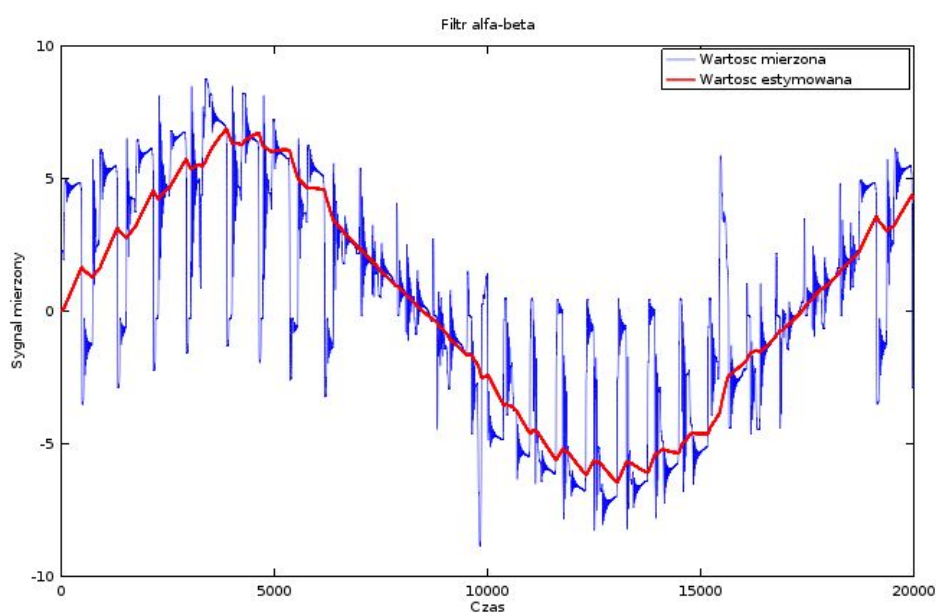
Budowanie układów zasilających górne bramki bez wykorzystania specjalistycznych układów, dla napięć wyższych niż 18V, nie jest prawdopodobnie najlepszym wyjściem. Użycie izolatora optycznego zwiększa rozmiar fizyczny jak i również cenę układu. Taka konstrukcja jest jednak odpowiednia dla niższych napięć. Dodatkowo, samodzielne tworzenie koncepcji takiego układu zwiększa szansę na popełnienie błędu. Sterowanie dolnych bramek, ponieważ używa układów scalonych, nie sprawiało żadnych problemów. Dobrą decyzją było też zasilanie dolnych bramek z układem przeciwnym, gdyż stosunkowo wysokie prądy bramki tranzystora dolnego, umożliwiając zwiększenie częstotliwości kluczowania napięcia, co okazało się potrzebne dla polepszenia wykrywania przejścia przez zero. Częstotliwość w finalnej wersji programu wynosi nieco ponad 30 kHz, może jednak być zwiększona, szczególnie jeśli sterownik miałby zasilać szybsze silniki.

5.2 Filtry

Żaden z filtrów nie został finalnie użyty w algorytmie komutacji. Stało się to ze względu na brak potrzeby – odpowiednie ustawienie pomiarów w czasie eliminuje potrzebę używania jakiegokolwiek filtracji napięcie.

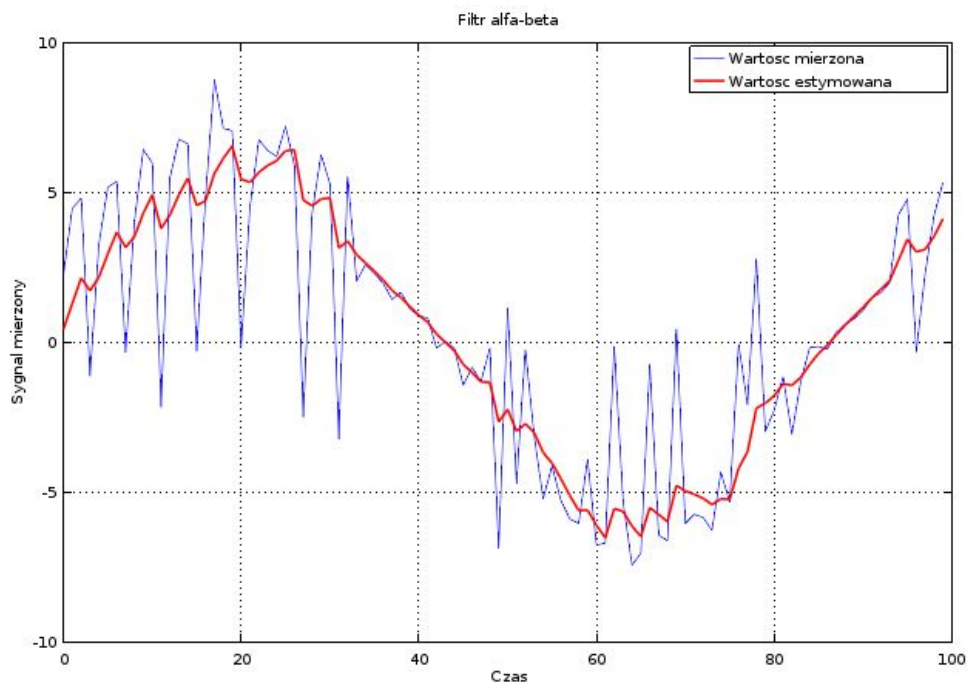
5.2.1 Filtr cyfrowy

W celu przetestowania skuteczności użyto programu Octave. Dane zostały pobrane z używanej wcześniej symulacji LT SPICE. Pobrano jeden okres sygnału pomiędzy punktem neutralnym z jedną fazą. Pierwotnie okres ma 20 tysięcy próbek. W każdej symulacji na jednostkę czasu przypadała jedna próbka.



Rysunek 52: Filtr alfa-beta, 20 tysięcy próbek

Jednak tak duża ilość próbek na jeden okres sygnału, który oryginalnie miał 800 Hz, nie jest fizycznie wykonalna. Dlatego zmniejszono ilość próbek do 100, wybierając co dwusetną próbkę z oryginalnego zestawu danych. Następnie znów eksperymentalnie nastrojono filtr.

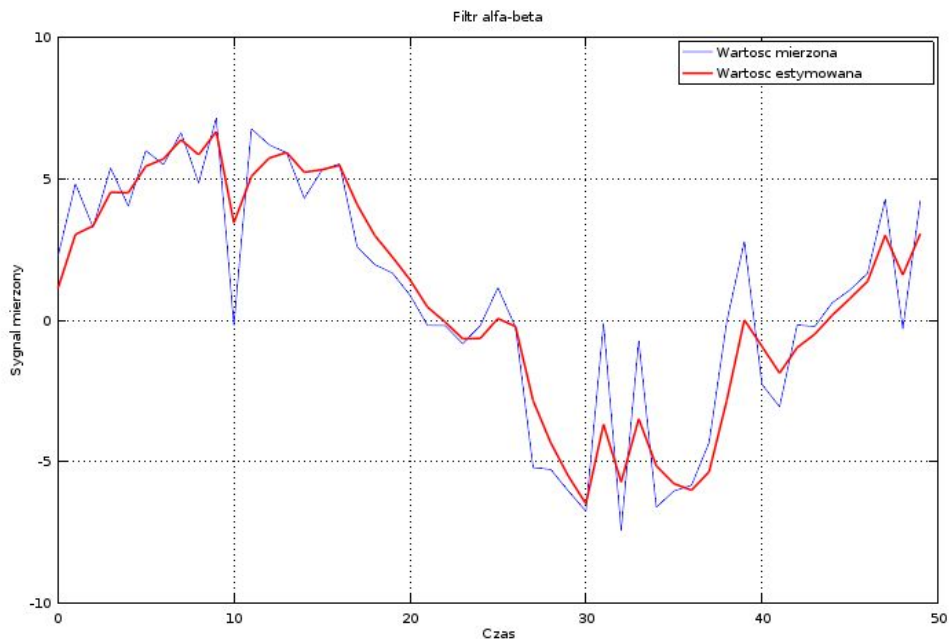


Rysunek 53: Filtr alfa-beta, 100 próbek

Taki zestaw danych mógłby zostać wygenerowany w sterowniku, gdyby konwersje ADC wywoływać regularnie co nieco ponad $1\ \mu\text{s}$. Wynik taki z pewnością umożliwiłby bardzo precyzyjną komutację.

Jednak pomiar i wywołanie kodu filtra co $1\ \mu\text{s}$ nadal jest stosunkowo częste, na tyle, że istnieją wątpliwości co do wykonalności takiego zadania. Następnie zmniejszono dalej liczbę próbek do 50, symulując tym samym albo sygnał o dwukrotnie większej częstotliwości, albo dwukrotnie zmniejszoną częstotliwość wywołania.

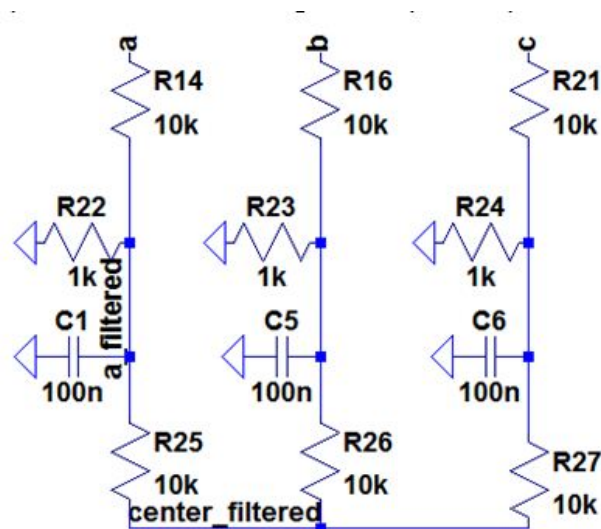
W tym wypadku, nastrojenie filtra jest znacznie trudniejsze, a pomimo tego należy się liczyć z niedokładnością estymacji. Przy tak małej ilości próbek filtr pozostanie podatny na zakłócenia.



Rysunek 54: Filtr alfa-beta, 50 próbek

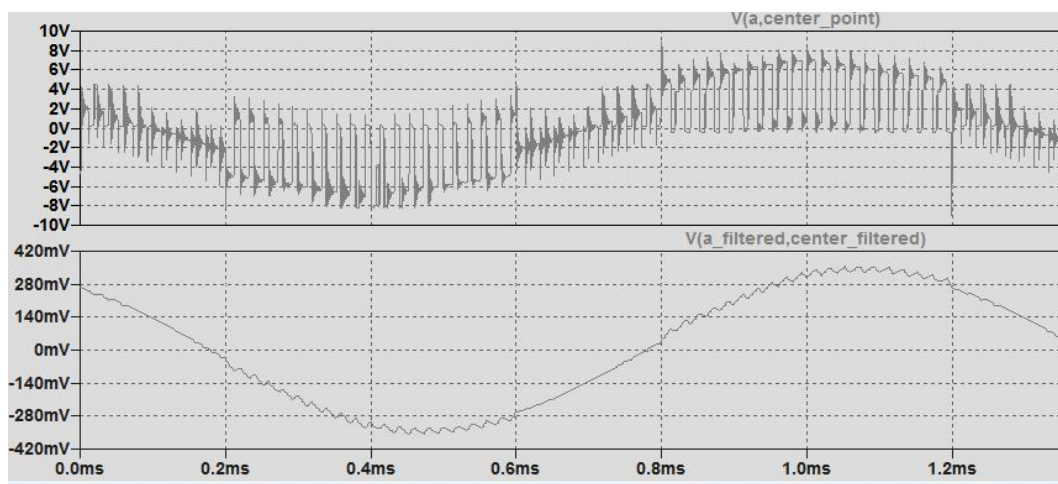
5.2.2 Filtr RC

Prosty filtr RC zbudowano zgodnie z poniższym schematem:



Rysunek 55: schemat elektryczna filtracji napięcia 3 faz silnika

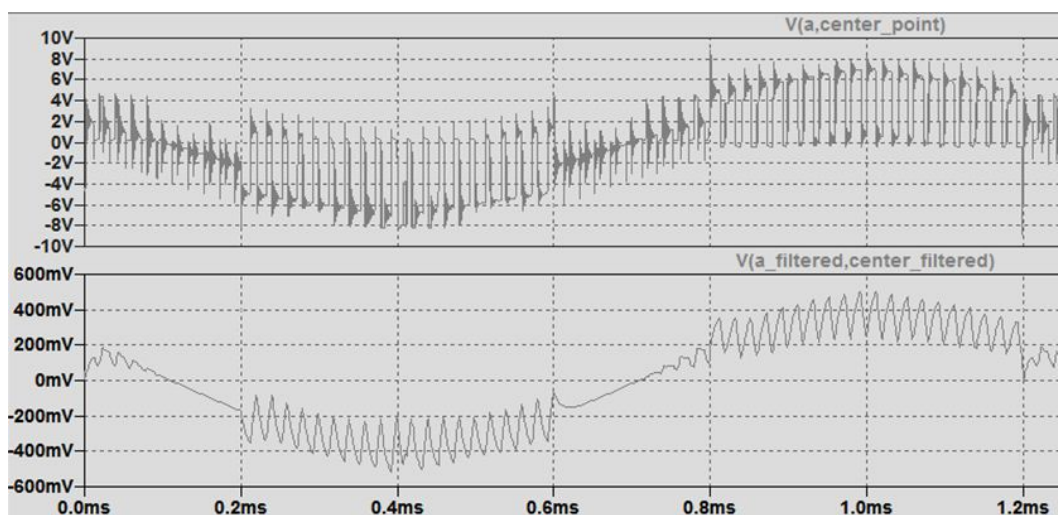
Częstotliwość graniczna dla takiego filtra wynosi około 1700 Hz. Układ został zbudowany fizycznie, jednak wyniki zostaną zaprezentowane w symulacji, ze względu na większą czytelność wykresów.



Rysunek 56: przebieg fazy silnika przed i po filtracji, filtrem RC z kondensatorami 100 nF

Parametry symulacji pozostały niezmienione w stosunku do poprzedniego rozdziału, a więc częstotliwość elektryczna sygnału wynosi 833 Hz. Na wykresie możemy zauważyć znaczne przesunięcie fazowe. Żeby sygnał był użyteczny, należałoby to przesunięcie w jakiś sposób skompensować wewnątrz programu.

Następnie zwiększono znacznie częstotliwość odcięcia, poprzez zmianę pojemności kondensatorów ze 100 nF na 10 nF. Oznacza to częstotliwość graniczną równą około 17 kHz.



Rysunek 57: przebieg fazy silnika przed i po filtracji, filtrem RC z kondensatorami 10 nF

Jak widać, taki filtr wystarcza, by pozbyć się zakłóceń o bardzo wysokiej częstotliwości, a przy tym generuje bardzo nieduże przesunięcie fazowe. W czasie kroku, trwającego na wykresie od 0.6 do 0.8ms, przesunięcie w czasie przejście przez zero

wynosiło tylko 13 μ s, co jest wynikiem bardzo dobrym, a przy tym czas ten jest łatwy do skompensowania w programie.

Wnioski:

Filtry RC są bardzo skuteczne, jeśli istnieje duża różnica pomiędzy częstotliwością odfiltrowywaną a właściwym sygnałem. Jeśli częstotliwość odcięcia filtra jest bliska częstotliwości pożądanego przebiegu, pojawia się kłopotliwe przesunięcie fazowe. Filtr tego rodzaju jest w stanie usunąć z sygnału zakłócenia związane z włączaniem i zamykaniem innych tranzystorów w układzie i może być używany w sterownikach silników BLDC, szczególnie jeśli przejście przez zero wykrywane jest z użyciem komparatora.

Filtr cyfrowy daje bardzo dobre rezultaty, a przy tym nie wprowadza opóźnień fazowe, co ułatwia komutację. Byłby przydatny w systemie, w którym nie da się przewidzieć momentu wystąpienia zakłóceń, lub tam gdzie są one ciągłe. Nadaje też do sygnałów nieco wolniejszych niż 1 kHz, tak by istniała wystarczająco ilość próbek na jeden okres. Natomiast w tym przypadku, równie dobre rezultaty osiągniemy odpowiednio pozycjonując pomiar, nie ma więc sensu implementować wymagającego obliczeniowo algorytmu.

5.3 Skuteczność komutacji

W tym podrozdziale zbadano i wyciągnięto wnioski z zachowania silnika podczas pracy. Przebiegi parametrów z testu zostały wykonane dzięki możliwości odczytywania zmiennych przez debugger przez przerywanie pracy mikrokontrolera.

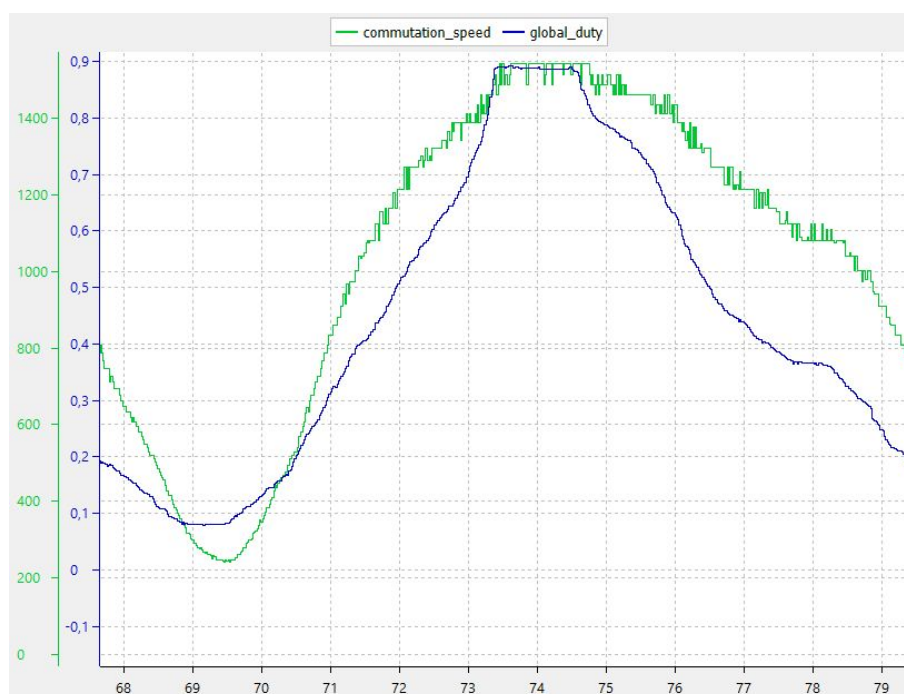
5.3.1 Komutacja na podstawie wykrywania przejść przez zero

Ze względu na przyjęty sposób wykrywania przejścia przez zero, istnieją wyraźne ograniczenia do co maksymalnej prędkości obrotowej silnika. Jeśli wykonywany byłby jeden pomiar na okres kluczowania prądu, na przykład z częstotliwością 20 kHz, pomiar wykonywany byłby tylko raz na 50 us. Tymczasem, dla silnika kręcącego się z prędkością 10 tys/Obr/min, jeden krok sekwencji zasilania trwa 143 us. W takiej sytuacji, pomiar wykona się jedynie dwa razy w ciągu kroku, co w oczywisty sposób nie zapewni dobrej komutacji. Pierwotnie przetestowano algorytm, który wywoływał kolejne konwersje przez cały okres trwania stanu wysokiego. Dla dużego wypełnienia PWM wywoływane było wiele konwersji, ale w momencie kiedy obniżano wypełnienie PWM, dokładność wykrywania spadała, podczas gdy silnik przez pewien czas nadal zachowywał wysoką prędkość. Sprawiało to, że komutacja nie była zbyt skuteczna. Silnik nie wypadał z synchronizmu, jednak fazy były źle zasilane, co było od razu słyszalne w dźwięku wydawanym przez silnik.

Dlatego zwiększono częstotliwość kluczowania prądu do ponad 30 kHz, a pomiary

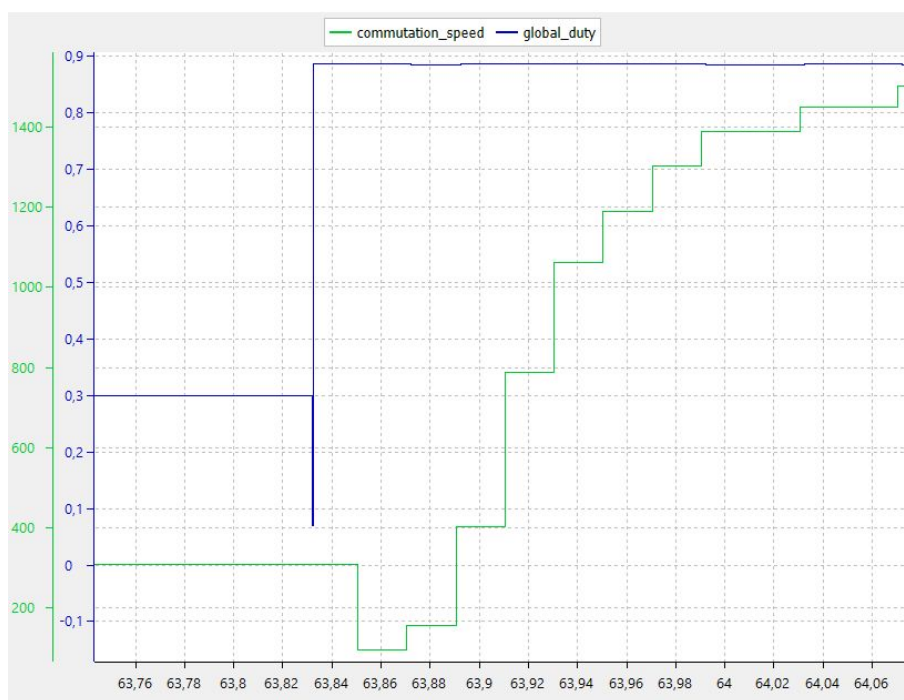
skonfigurowano tak, by prawie zawsze wywoływały się dwie konwersje w ciągu jednego okresu PWM. Inaczej jest jedynie dla niskich wypełnień sygnału (mniej niż kilkanaście procent), co jednak nie wydaje się sprawiać problemów. Obecny algorytm zasila stabilnie silnik kręcący się z prędkością 14 tys. Obr/min. Należy się spodziewać, że przy wyższych prędkościach problemy pojawiłyby się ponownie, jednak obecnie uzyskiwane prędkości są wystarczające dla większości zastosowań w robotach latających. Należy pamiętać, że silnik z obciążeniem obraca się wolniej niż by to wynikało ze stałej K_V (przy dużym obciążeniu może to być $0.6-0.7 K_V$)

Wykonano wiele zapisów parametrów pracy silnika, które zostaną przedstawione poniżej, w celu zaprezentowania skuteczności działania algorytmu.



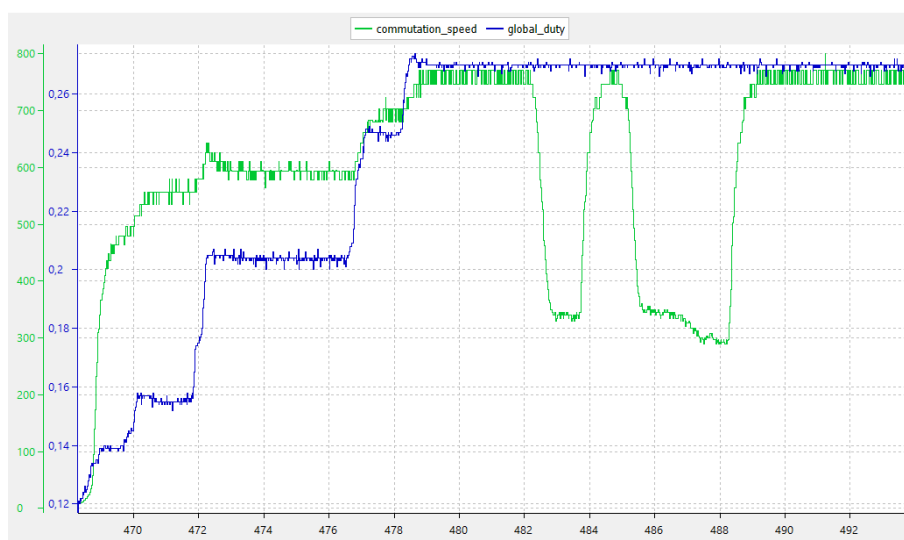
Rysunek 58: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji elektrycznej, przy zadawaniu wypełnienia w aparaturze modelarskiej

Zielony wykres zawiera informację o częstotliwości obrotów elektrycznych w hercach, natomiast niebieski oznacza wypełnienie sygnału kluczującego, w zakresie od 0 do 1. Na osi poziomej znajduje się czas w sekundach. Na każdym z kolejnych wykresów kolory te będą miały takie samo znaczenie. Wypełnienie było zmieniane ręcznie, poprzez dżędek na aparaturze modelarskiej. Widać jak prędkość silnika zmienia się w zależności od zadanego wypełnienia. Widać też, że im wyższa częstotliwość komutacji, tym mniej dokładny jest pomiar prędkości.



Rysunek 59: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji elektrycznej podczas rozpędzania silnika

Następnie zmierzono szybkość rozpędzania się silnika, od całkowitego zatrzymania do pełnej prędkości obrotowej. Komutacja z wykrycie przejścia przez zero zaczyna się w momencie skoku wartości wypełnienia, wcześniej trwa rozruch. Niska rozdzielczość narysowanej krzywej wynika z faktu, że dane były przesyłane co 20 ms, a więc stosunkowo rzadko, jak na tak szybkie zjawiska.



Rysunek 60: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji elektrycznej podczas obciążania silnika

Następnie sprawdzono jak zachowuje się silnik po dołożeniu obciążenia, bez zmiany wypełnienia PWM. W 428. sekundzie (w połowie wykresu) wał silnika został ściśnięty z użyciem okładzin, tak by tarcie hamowało silnik. Następnie obciążenie zdjęto i przyłożono ponownie. Efektem tego były znaczne spadki prędkości obrotowej silnika. Przez dokładanie obciążenia sprawdzowano też minimalną częstotliwość elektryczną, przy której komutacja działa. Dodkładano obciążenie przy stałym wypełnieniu PWM. Błąd odczytu prędkości i zatrzymanie pojawiło się około 45 Hz, czyli około 400 obr/min.



Rysunek 61: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji elektrycznej podczas testowania najmniejszej prędkości obrotowej, przy której działa komutacja

5.3.2 Rozruch silnika

Ponieważ algorytm bezczujnikowej komutacji nie zapewnia sprzężenia zwrotnego, dotyczącego pracy silnika podczas rozruchu, konieczne jest dobranie parametrów rozpędzania silnika do znanego obciążenia. Dlatego w czasie testów, sprawdzono, czy silnik skutecznie rozpędza się zarówno bez żadnego obciążenia, jak i z ośmiocalowym śmigłem. Wszystkie testy przebiegły pomyślnie. Zdarzają się sporadyczne błędy w fazie pomiędzy rozruchem a komutacją z wykryciem przejścia przez zero, nie powodują one jednak zatrzymania silnika. Rozpędzanie silnika trwa około 350 ms. Czas ten możemy zauważyć na przykład rys 66, w czasie kiedy wypełnienie jest równe 0.4.

Wnioski:

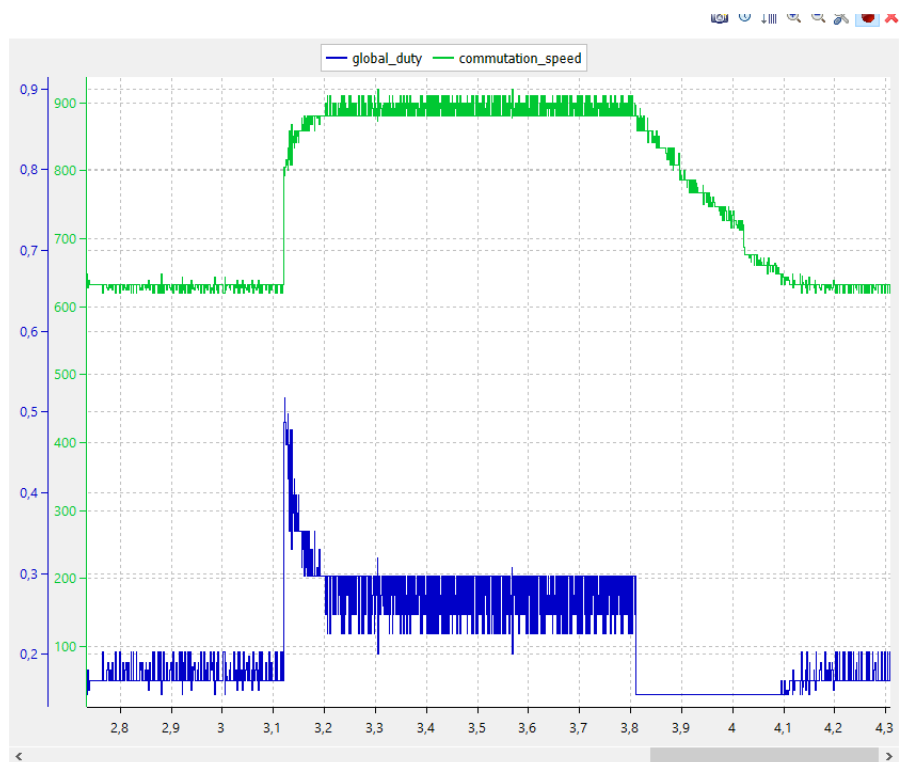
Komutacja działa bardzo dobrze. Występują sporadyczne błędy, najczęściej nie powodujące zatrzymania silnika. Silnik dobrze reaguje na zmiany wypełnienia sygnału PWM, nawet te bardzo szybkie.

5.4 Regulator prędkości obrotowej

Ponieważ stworzenie algorytmu regulacji prędkości było jedną z ważniejszych części pracy, poniżej przedstawiono szczegółowo testy skuteczności regulatora.

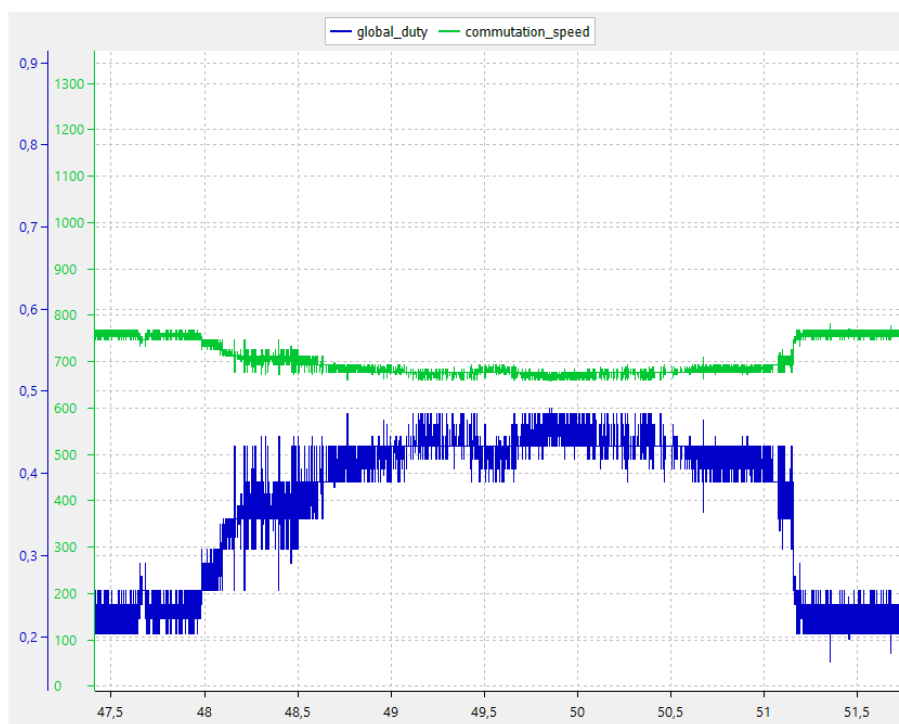
5.4.1 Regulator proporcjonalny

Najpierw zaimplementowano tylko regulator proporcjonalny, następnie przeprowadzono jego testy.



Rysunek 62: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, przy zmienianiu wartości zadanej pomiędzy 700 i 1000 Hz dla regulatora proporcjonalnego

Program testowy zadawał skokową częstotliwość elektryczną, raz 700 a raz 1000 Hz. Zmiana prędkości obrotowej jest dosyć szybka (od zmiany wartości na większą do ustalenia mija ok. 100 ms), jednak występuje znaczny uchyb ustalony (ok. 10 %).



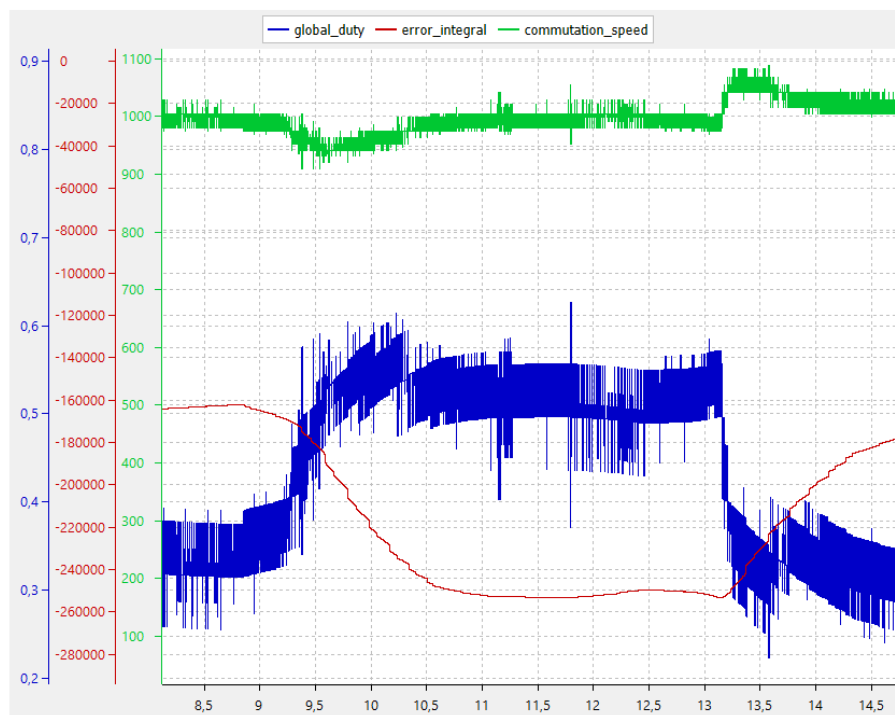
Rysunek 63: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, podczas dokładania i zdejmowania obciążenia ze stałą prędkością zadana dla regulatora proporcjonalnego

W kolejnym teście na silnik zostało dołożone, a następnie odjęte obciążenie. Widoczna jest zdecydowana reakcja regulatora, objawiająca się poprzez zwiększenie wypełnienia sygnału PWM. Uchyb ustalony jednak zwiększa się, czego należy się spodziewać po regulatorze proporcjonalnym.

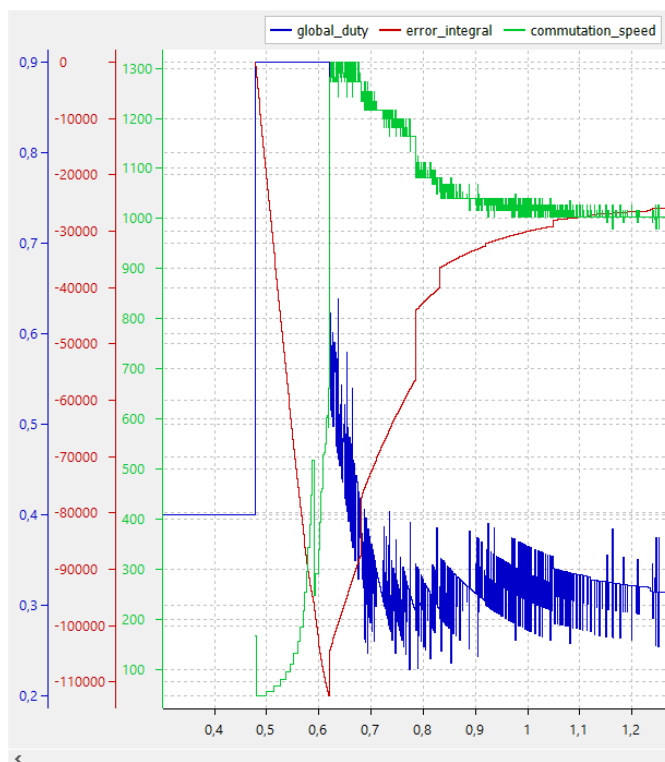
5.4.2 Regulator proporcjonalny i całkujący

Następnie, do regulatora wprowadzono człon całkujący i ponownie przeprowadzono testy. Zadana częstotliwość wynosiła 1000 Hz. Przed dołożeniem obciążenia widać (rys. 64), że wartość ustaliła się na tym poziomie, co oznacza, że uchyb ustalony nie występował. Po dołożeniu obciążenia widać niewielki spadek prędkości obrotowej, trwający do czasu, aż zareaguje człon całkujący. Czerwony wykres przedstawia wartość nacałkowanego błędu. Następnie wartość ponownie ustala się blisko wartości zadanej, pomimo stałego obciążenia. Po odjęciu obciążenia człon całkujący wywołuje niewielkie przekroczenie wartości zadanej.

Do tej pory wzmocnienia poszczególnych członów były ustawione dosyć zachowawczo, co widać po powolnych reakcjach, szczególnie członu całkującego. Po zwiększeniu jego wzmocnienia, pojawiło się dosyć mocno nasilone zjawisko wind-up, polegające na nagromadzeniu błędu przez człon całkujący, w czasie gdy wyjścia regulatora znajduje się w nasyceniu (zmienna sterująca osiągnęła ograniczenia).



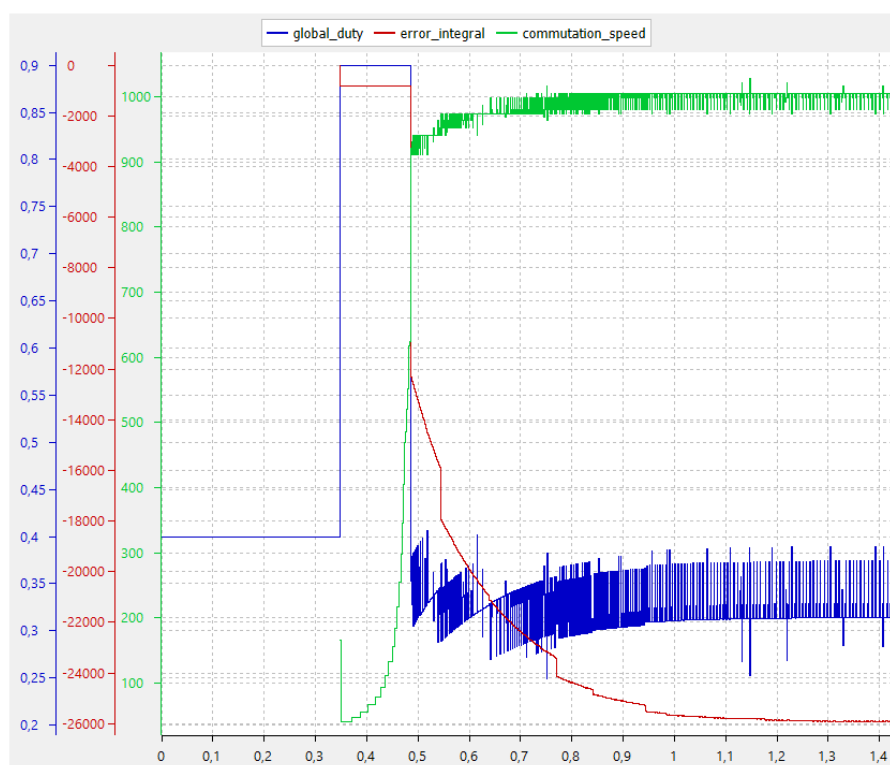
Rysunek 64: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, podczas dokładania i zdejmowania obciążenia ze stałą prędkością zadana dla regulatora proporcjonalno-całkującego



Rysunek 65: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, oraz wartości całkowanego błędu podczas rozruchu silnika, podczas występowania zjawiska wind-up

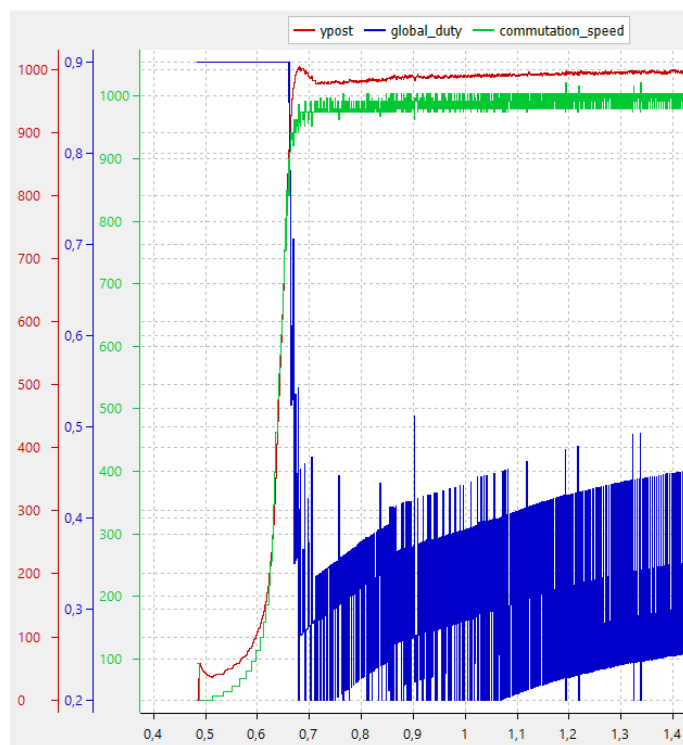
Widać, że w czasie rozpędzania, gdy wartość wypełnienia przez dłuższy czas ma maksymalną wartość, gromadzi się wartość całkowana, co w następstwie skutkuje dużym przesterowaniem. By zlikwidować to zjawisko do kodu dodano warunek, który wyłącza całkowanie, gdy wyjście regulator jest w nasyceniu.

Efektom tego jest działający algorytm anti-windup, który eliminuje przesterowanie po wyjściu z nasycenia.



Rysunek 66: przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, oraz wartości całkowanego błędu podczas rozruchu silnika, z działającym algorytmem anti-windup

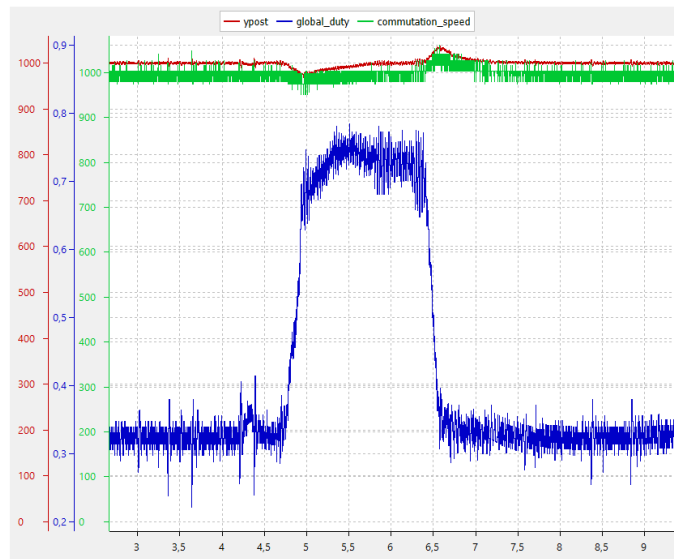
Po zwiększeniu wzmocnienie proporcjonalnego, uwidocznił się negatywny wpływ niedokładności pomiaru prędkości na odpowiedź regulatora. Skutkiem wahań w pomiarze prędkości są duże wahania na wyjściu regulatora. Dlatego sprawdzono, czy użycie opisanego wcześniej filtra alfa-beta nie pomoże w zmniejszeniu błędów odczytywanej prędkości. Na następnym wykresie wykresie widac że wartość wypełnienia zmienia się bardzo szybko i w szerokim zakresie. Było to na tyle intensywne, że było zauważalne poprzez mniej uporządkowany dźwięk, wydawany przez silnik. Jako następny krok postanowiono użyć przefiltrowanej prędkości jako wejścia regulatora.



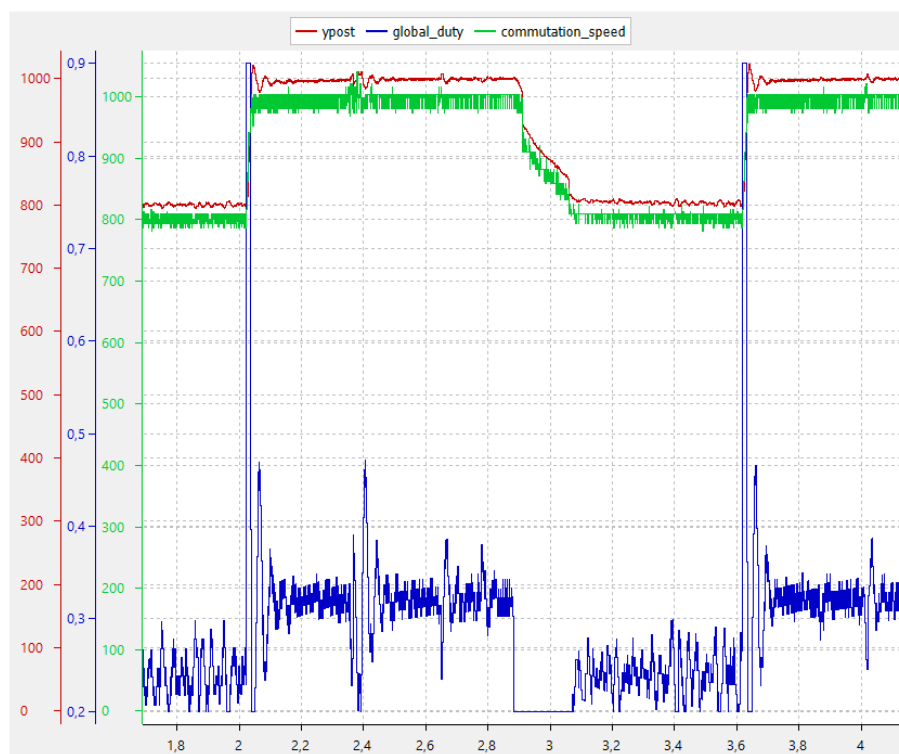
Rysunek 67: przebieg wypełnienia sygnału PWM, częstotliwości komutacji, oraz przefiltrowanej częstotliwości komutacji (czerwony wykres), z regulatorem korzystającym z niefiltrowanej częstotliwości

Skutkiem tego było znaczne zmniejszenie wahań wartości sterującej, co widać w przeprowadzonym następnie teście, polegającym na dokładaniu i ściąganiu obciążenia z pracującego silnika.

Jako ostatni test, gotowego już regulatora, program zadawał skokowo prędkość, raz 800 a raz 1000 Hz. Jak widać, wyjście filtru ma niestety pewną skłonność do oscylacji, co potem jest wzmacniane przez regulator prędkości. Na szczęście oscylacje te szybko gasną. Jest też prawdopodobne, że lepsze nastrojenie filtru pomogło by się tego pozbyć.



Rysunek 68: przebieg wypełnienia sygnału PWM, częstotliwości komutacji, oraz przefiltrowanej częstotliwości komutacji (czerwony wykres), z regulatorem korzystającym z filtrowanej częstotliwości, podczas dokładania i ściągania obciążenia



Rysunek 69: przebieg wypełnienia sygnału PWM, częstotliwości komutacji, oraz przefiltrowanej częstotliwości komutacji (czerwony wykres), z regulatorem korzystającym z filtrowanej częstotliwości, podczas zmieniania zadanej częstotliwości pomiędzy 800 a 1000 Hz

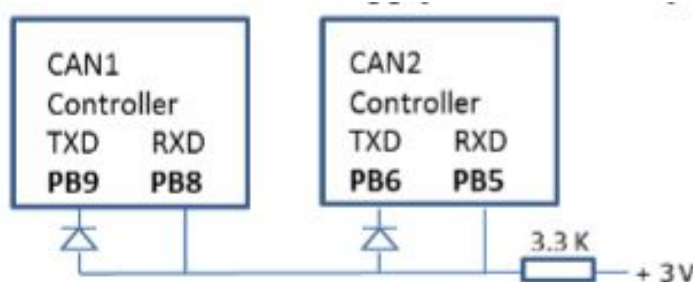
Wnioski:

Uzyskano funkcjonalny regulator, który nawet przy szybkich i znacznych zmianach obciążenia potrafi utrzymać zadaną prędkość (na rys 68 maksymalny błąd prędkości wynosił około 3%). Zastosowano algorytm przeciwdziałający zjawisku wind-up, oraz udowodniono przydatność filtra alfa-beta w polepszaniu działania regulatora prędkości.

5.5 Komunikacja CAN

By przetestować działanie wysyłania danych po magistrali CAN, należy przygotować urządzenie, zdolne ten sygnał odebrać. Jednak w mikrokontroler wbudowany jest jedynie kontroler komunikacji, nie ma w nim układu odbiorczo-nadawczego, pozwalającego przenieść komunikację na poziom fizyczny (dotyczy to na przykład używania odpowiednich napięć i natężeń prądu). Była to przeszkoda w wykonaniu testów, głównie ze względu ograniczony dostęp do uczelni i jej pracowników, związany z aktualnie trwającą epidemią.

Jednak w nocy aplikacyjnej *CAN Primer: Creating Your Own Network*[12] udało się znaleźć sposób na stworzenie sieci testowej, wykorzystującej dwa kontrolery CAN, znajdujące się wewnątrz mikrokontrolera STM32F407. W sieci testowej istnieją ograniczenia co do prędkości transmisji, nie należy się też spodziewać odporności na zakłócenia, pozwala ona jednak sprawdzić poprawność działania programu. Należy połączyć dwa kontrolery CAN w sposób przedstawiony na schemacie, używając dwóch diod sygnałowych.



Rysunek 70: Przedstawienie sposobu połączenia dwóch sterowników CAN, w celu utworzenia sieci testowej, pochodzi z [12]

Jednak ze względu na fakt, że rozwiązanie problemu udało się odnaleźć stosunkowo późno, testów komunikacji nie wykonano.

5.6 Podsumowanie wykonania celów pracy

W tym podrozdziale zostaną zwięźle podsumowane wyniki prac, w odniesieniu do celów i założeń, postawionych na początku projektu.

5.6.1 Cele pracy

Zaprojektowano kompletny regulator obrotów silnika BLDC, korzystając jedynie z gotowej płytki prototypowej z osadzonym na niej mikrokontrolerem. Pozostałe potrzebne części funkcjonalne zostały zaprojektowane i samodzielnie zmontowane. Sprawdzone poprawność funkcjonowania tej części projektu i nie znaleziono żadnych istotnych uchybień, uniemożliwiających realizację celów pracy.

Stworzono algorytm, który powoduje skuteczne zasilanie silnika, zarówno jego normalną komutację z użyciem sprzężenia zwrotnego jak i rozruch. Działające podstawowe elementy umożliwiły realizację nieco bardziej zaawansowanego zadania - regulację prędkości obrotowej.

Udało się skutecznie odbierać i interpretować sygnał PWM z odbiornika modelarskiego. Jedynym celem, którego działania nie uwodniono, jest komunikacja po magistrali CAN. Zostały stworzone programowe podstawy do używania tej magistrali, nie potwierdzono jednak ich działania poprzez praktyczne testy.

Dodatkowo, w celu pokazania szerszych możliwości mikrokontrolera, zbadano możliwość cyfrowego filtrowania napięć faz silnika. Technika ta okazała się nieprzydatna dla specyficznych warunków pracy sterownika (współpraca z bardzo szybkimi silnikami), jest jednak obiecująca w ogólnym kontekście regulacji obrotów silnika BLDC, ponieważ udało się ją wykorzystać do filtracji odczytów prędkości.

5.6.2 Założenia projektowe

Udało się zaprojektować układ elektryczny, zgodny z wcześniej postanowionymi założeniami. Z symulacji wynika, że stworzony układ jest w stanie przewodzić prąd o natężeniu 30A w szynie prądu stałego. Nie są też wykorzystywane żadne wyspecjalizowane układy scalone - użyte zostały tylko tranzystory, uniwersalne układy czasowe NE555 oraz inne powszechnie używanego, podstawowe elementy elektroniczne. Wykonany fizycznie układ nie spełnia założenia dotyczącego zasilania wyższymi napięciami, jednak błąd ten został naprawiony i przedstawiono jego rozwiązanie. Dodatkowo, sprawdzono użyteczność dwóch rodzajów filtracji napięcia faz silnika.

Spis ilustracji

Rysunek 1	Uproszczona budowa silnika BLDC	7
Rysunek 2	Uprozczone przebiegi faz silnika, rysowane względem punktu neutralnego silnika	8
Rysunek 3	Odpowiednio: przebiegi napięcia fazy A względem uziemienia, napięcie fazy A względem punktu neutralnego, sygnał sterujący tranzystorem kluczującym prąd	11
Rysunek 4	Napięcie fazy zasilanego silnika przy wypełnieniu PWM równym 100%. Pokazuje skoki napięcia pojawiające się przy zmianie kroku	12
Rysunek 5	Typowa konfiguracja trójfazowego mostka H	13
Rysunek 6	porównanie technologii tranzystorów wysokiej mocy. Opis: BJT: wysokie napięcie, niski prąd, MOSFET: wysoki prąd, niskie napięcie, IGBT: wysokie napięcie, wysoki prąd	15
Rysunek 7	Zależność napięcia nasycenia od prądu kolektora	16
Rysunek 8	Uproszczony schemat zasilania silnika	17
Rysunek 9	Część karty katalogowej IRLR8743	18
Rysunek 10	zależność pomiędzy rezystancją przewodzenia a temperaturą złącza tranzystora karty katalogowej IRLR8743	19
Rysunek 11	Symbol tranzystora z uwzględnieniem wewnętrznej diody z karty katalogowej IRLR8743	20
Rysunek 12	zależność pomiędzy natężeniem prądu a spadkiem na diodzie wewnętrznej tranzystora z karty katalogowej IRLR8743	21
Rysunek 13	Główna część układu symulacyjnego w LTSPICE	22
Rysunek 14	Część układu symulująca pracujący silnik	23
Rysunek 15	Przebieg prądu drenu górnego tranzystora	23
Rysunek 16	Zestawienie sygnału sterującego(górny wykres), wydzielanej mocy(środkowy wykres) oraz prądu drenu górnego tranzystora	23
Rysunek 17	Schemat otoczenia górnego tranzystora, z dodatkową diodą	24
Rysunek 18	Przebieg prądu drenu dolnego tranzystora	25
Rysunek 19	Zestawienie sygnału sterującego, wydzielanej mocy i prądu drenu dolnego tranzystora	25
Rysunek 20	Schemat otoczenia dolnego tranzystora z dodatkową diodą	25
Rysunek 21	Przebieg prądu tranzystora i dodatkowej diody	25
Rysunek 22	Zestawienie sygnału sterującego i ciepła wydzielanego na dolnym tranzystorze	26
Rysunek 23	Przykładowy układ sterujący bramką z AND9093/D	27
Rysunek 24	Schemat układu sterowania bramką (GATE AH) sterownika BlueESC	28

Rysunek 25	Uproszczony układ symulacyjny ze sterowaniem górnej bramki analogicznym do BlueESC	30
Rysunek 26	Przebieg napięcia pomiędzy bramką a drenem tranzystora górnego dla układu jak na rys. 24	30
Rysunek 27	Układ sterujący górną bramką, ze źródłem napięciowym zamiast układu bootstrap	30
Rysunek 28	Przebieg napięcia pomiędzy bramką a drenem tranzystora górnego dla układu jak na rys. 27	31
Rysunek 29	Układ sterujący górną bramką, ze źródłem napięciowym i odniesiem tranzystora Q1 do fazy	31
Rysunek 30	Przebieg napięcia pomiędzy bramką a drenem tranzystora górnego dla układu jak na rys. 29	31
Rysunek 31	Układ sterujący górną bramką, ze źródłem napięciowym i izolacją optyczną wejścia cyfrowego	33
Rysunek 32	Przebieg napięcia pomiędzy bramką a drenem tranzystora górnego po zastosowaniu izolacji optycznej wejścia	33
Rysunek 33	Układ sterowania bramką dolną, z przeciwobnymi tranzystorami bipolarnymi NPN	34
Rysunek 34	Przebiegi prądów bramki, napięcia sterującego i napięcia bramki, jak dla układu na rys.33	35
Rysunek 35	Uproszczony schemat działania układu NE555 z karty katalogowej	35
Rysunek 36	Schemat sterowania bramką dolną, z wykorzystaniem układu NE555	36
Rysunek 37	Zestawienie prądu bramki, sygnału sterującego i napięcia bramki tranzystora M1	36
Rysunek 38	Schemat układu generującego wyższe napięcie, składającego się z NE555 oraz pompy ładunkowej	37
Rysunek 39	Przebiegi napięć wyjścia OUT NE555(n004), wyjścia pompy ładunkowej(Vh+), niższego napięcia, zasilającego NE555(Vreg) oraz zasilania całego sterownika (VCC)	38
Rysunek 40	Schemat elektryczny dzielników napięcia i sztucznego punktu neutralnego. Wyjściami układu są węzły PHASE_x_O (gdzie x to oznaczenie fazy, A, B lub C) oraz center, wejściami punkty phase_x	39
Rysunek 41	Umieszczenie cewki symulującej pasożtyniczną pojemność na schemacie układu	40
Rysunek 42	Zestawienie sygnału sterującego tranzystorem CH i napięcia na źródłach tranzystorów górnych, z widocznymi indukcyjnymi skokami	40
Rysunek 43	Umieszczenie kondensatora filtrującego na schemacie układu	41

Rysunek 44	Zestawienie sygnału sterującego tranzystorem CH i napięcia na źródłach tranzystorów górnych, po dodaniu kondensatora filtrującego	41
Rysunek 45	Zestawienie sygnału sterującego tranzystorem CH i napięcia na źródłach tranzystorów górnych, po dodaniu kondensatora filtrującego, z uwzględnieniem jego dodatkowej indukcyjności	42
Rysunek 46	Krzywe ułatwiające dobór grubości miedzianej ścieżki PCB. Na osi pionowej - natężenie prądu w amperach, na poziomej - pole przekroju ścieżki w mil^2 , różne krzywe zostały narysowane dla różnych wzrostów temperatury ścieżki.	43
Rysunek 47	Schemat płytki PCB z podziałem na funkcjonalne części . .	45
Rysunek 48	Wizualizacja wykonanej płytki PCB. Z lewej znajduje się strona przednia, z prawej spodnia	45
Rysunek 49	Zdjęcie wykonanej płytki PCB	46
Rysunek 50	Wizualizacja drugiej wersji płytki	47
Rysunek 51	Rysunek przebiegu typowego dla modeli latających sygnału PWM	58
Rysunek 52	Filtr alfa-beta, 20 tysięcy próbek	64
Rysunek 53	Filtr alfa-beta, 100 próbek	65
Rysunek 54	Filtr alfa-beta, 50 próbek	66
Rysunek 55	schemat elektryczna filtracji napięcia 3 faz silnika	66
Rysunek 56	przebieg fazy silnika przed i po filtracji, filtrem RC z kondensatorami 100 nF	67
Rysunek 57	przebieg fazy silnika przed i po filtracji, filtrem RC z kondensatorami 10 nF	67
Rysunek 58	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji elektrycznej, przy zadawaniu wypełnienia w aparatury modelarskiej	69
Rysunek 59	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji elektrycznej podczas rozpędzania silnika	70
Rysunek 60	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji elektrycznej podczas obciążania silnika	70
Rysunek 61	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji elektrycznej podczas testowania najmniejszej prędkości obrotowej, przy której działa komutacja	71
Rysunek 62	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, przy zmienianiu wartości zadanej pomiędzy 700 i 1000 Hz dla regulatora proporcjonalnego	72
Rysunek 63	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, podczas dokładania i zdejmowania obciążania ze stałą prędkością zadana dla regulatora proporcjonalnego	73

Rysunek 64	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, podczas dokładania i zdejmowania obciążenia ze stałą predkością zadana dla regulatora proporcjonalno-całkującego	74
Rysunek 65	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, oraz wartości całkowanego błędu podczas rozruchu silnika, podczas występowania zjawiska wind-up	74
Rysunek 66	przebieg wypełnienia sygnału PWM oraz częstotliwości komutacji, oraz wartości całkowanego błędu podczas rozruchu silnika, z działającym algorytmem anti-windup	75
Rysunek 67	przebieg wypełnienia sygnału PWM, częstotliwości komutacji, oraz przefiltrowanej częstotliwości komutacji (czerwony wykres), z regulatorem korzystającym z niefiltrowanej częstotliwości	76
Rysunek 68	przebieg wypełnienia sygnału PWM, częstotliwości komutacji, oraz przefiltrowanej częstotliwości komutacji (czerwony wykres), z regulatorem korzystającym z filtrowanej częstotliwości, podczas dokładania i ściągania obciążenia	77
Rysunek 69	przebieg wypełnienia sygnału PWM, częstotliwości komutacji, oraz przefiltrowanej częstotliwości komutacji (czerwony wykres), z regulatorem korzystającym z filtrowanej częstotliwości, podczas zmiany zadanej częstotliwości pomiędzy 800 a 1000 Hz	77
Rysunek 70	Przedstawienie sposobu połączenia dwóch sterowników CAN, w celu utworzenia sieci testowej, pochodzi z [12]	78

Literatura

- [1] | *AVR444: Sensorless control of 3-phase brushless DC motors*, nota aplikacyjna, Atmel
<http://ww1.microchip.com/downloads/en/AppNotes/doc8012.pdf> dostęp online 03.2020
- [2] | *AN857: Brushless DC Motor Control Made Easy*, nota aplikacyjna, W. Brown, Microchip Technology Inc.,
<http://ww1.microchip.com/downloads/en/appnotes/00857b.pdf> dostęp online 03.2020
- [3] | *AN1914: 3-Phase BLDC Motor Control with Sensorless Back EMF Zero Crossing Detection Using 56F80x*, nota aplikacyjna, L. Prokop, L. Chalupa, Freescale Semiconductors
<https://www.nxp.com/docs/en/application-note/AN1914.pdf> dostęp online 03.2020
- [4] | *UM1594: Six-step sensorless BLDC motor drive example software for the STM32*, instrukcja, STMicroelectronics
- [5] | *BlueESC Documentation*, dokumentacja sterownika, BlueRobotics
<http://docs.bluerobotics.com/bluesc/> dostęp online 03.2020
- [6] | *FET vs. BJT vs. IGBT: What's the Right Choice for Your Power Stage Design?*, artykuł, L. Mays
<https://www.allaboutcircuits.com/technical-articles/fet-vs-bjt-vs-igbt-whats-the-right-choice-for-your-power-stage-design/> online 03.2020
- [7] | *AND9093/D: Using MOSFETs in Load Switch Applications*, nota aplikacyjna, ON Semiconductors
<https://www.onsemi.com/pub/Collateral/AND9093-D.PDF> dostęp online 03.2020
- [8] | *Engine Speed Monitoring: The Alpha-Beta Filter*, artykuł, Microstar Laboratories, Inc.
<http://www.mstarlabs.com/control/engspeed.html>
- [9] | *Elektroniczny komutator silnika BLDC* praca inżynierska, T. Kostur, W. Nowakowski, Poznań 2014
- [10] | *Filtr Alfa – Beta od teorii do praktyki – #1*, artykuł (blog)
<https://forbot.pl/blog/filtr-alfa-beta-teorii-praktyki-1-id2234> dostęp online 03.2020
- [11] | *Kurs STM32 F4*, artykuł (blog)
<https://forbot.pl/blog/kurs-stm32-f4-1-czas-poznac-hal-spis-tresci-kursu-id14114> dostęp online 03.2020

- [12]] *CAN Primer: Creating Your Own Network*, nota aplikacyjna, ARM Ltd.
http://www.keil.com/appnotes/files/apnt_236.pdf dostęp online 03.2020
- [13]] *xx555 Precision Timers*, karta katalogowa, Texas Instruments
<http://www.ti.com/lit/ds/symmlink/ne555.pdf> dostęp obline 03.2020
- [14]] *IPC-2221A: Generic Standard on Printed Board Design*, norma, IPC Association Connecting Electronics Industries
[http://www-eng.lbl.gov/shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A\(L\).pdf](http://www-eng.lbl.gov/shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A(L).pdf) dostęp online 03.2020
- [15]] *AN047: Brushless DC Motor Fundamentals*, nota aplikacyjna, Jian Zhao/Y-angwei Yu, Monolithic Power Systems, Inc.
https://www.monolithicpower.com/pub/media/document/Brushless_DC_Motor_Fundamentals.pdf dostęp online 03.2020
- [16]] *Current Carrying Capacity of Vias*, artykuł, D. Brooks, D. Graves, UltraCAD
<https://www.ultracad.com/articles/viacurrents.pdf> dostęp online 03.2020

Zawartość CD

- Praca dyplomowa w postaci źródłowej
- Praca dyplomowa w postaci pliku pdf
- Kod źródłowy oprogramowania
- Schemat elektryczny